Generating Explanations of Robot Policies in Continuous State Spaces

Oliver Struckmeier

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 10.8.2018

Supervisor

Prof. Ville Kyrki
Prof. George Nikolakopoulos

Advisor

M.Sc. Mattia Racca

Aalto University
School of Electrical
Engineering

Aalto University
School of Electrical
Engineering

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi
Abstract of the master's thesis

Author Oliver Struckmeier

Title Generating Explanations of Robot Policies in Continuous State Spaces

Degree programme Erasmus Mundus Master in Space Science and Technology

Major Space Robotics and Automation **Code of major** ELEC3047

Supervisor Prof. Ville Kyrki
Prof. George Nikolakopoulos

Advisor M.Sc. Mattia Racca

Date 10.8.2018 **Number of pages** 56+4 **Language** English

Abstract

Transparency in HRI describes the method of making the current state of a robot or intelligent agent understandable to a human user. Applying transparency mechanisms to robots improves the quality of interaction as well as the user experience.

Explanations are an effective way to make a robot's decision making transparent. We introduce a framework that uses natural language labels attached to a region in the continuous state space of the robot to automatically generate local explanations of a robot's policy.

We conducted a pilot study and investigated how the generated explanations helped users to understand and reproduce a robot policy in a debugging scenario.

Keywords Robotics, Human-Robot Interaction, Transparency, Mental Models, Explanations

Preface

This thesis was written in the Intelligent Robotics research group at Aalto University in Finland during my final year of the SpaceMaster program. It has been an honor and a great pleasure to work in such an inspiring and positive environment and I want to hereby thank my supervisor Prof. Ville Kyrki for following my work and always providing advice and help. I also want to thank my instructor Mattia Racca for his valuable advice and positive attitude that ignited my passion for research and academia.

I want to thank my colleagues for taking part in the user study of this thesis and for providing valuable input for further work. Furthermore, I want to thank my SpaceMaster colleagues here at Aalto for sticking together, sharing information and helping each other out whenever possible.

This thesis is dedicated to my parents. Without their support throughout my educational career this would not have been possible and I am most grateful for being blessed with such amazing parents.

Espoo, 10.8.2018

Oliver Struckmeier

Contents

List of Figures

List of Symbols

A set of actions

a robot action

c concept

$D(P, Q)$ Kullback-Leibler Divergence

d sum of values v that have the same concept c

$d_p(P, Q)$ Jensen-Shannon Distance between two probability distributions P and Q

$h_{a,p}$ histogram of the number of samples with the same action a and parameter p

J number of concept in a parameter p

$JS(A, B)$ Jaccard Similarity of two finite sets A and B

$JSD(P, Q)$ Jensen-Shannon Divergence between two probability distributions P and Q

M number of actions

m_c scale ratio for a concept c

N number of parameters

n sum of values v of samples that have the same action a and concept c

P set of all parameters of the state space S

p parameter

r concept ratio of concept c

r_c^* scaled concept ratio of concept c

r_p performance as measured in correctly labeled samples to all samples

S state space

s current robot state in the state space S

$\overline{s}$ correctly labeled samples

S_c number of samples in a concept c

s' next robot state after taking action a

s_t state in the state space that is to be explained

$scale_p$ scale value to weight the parameters calculated using the averaged Jensen-Shannon distances between all histograms $h_{a,p}$

t one tiles in the grid world of the experiment

T set of all tiles t in the grid world of the experiment

v_c concept strength for a concept c

x value of a parameter p, $x \in p$

$\gamma(s)$ concept of a sample state s

μ mean

$\mathcal{N}(\mu, \sigma)$ Gaussian Normal Distribution

π Policy

σ standard deviation

σ_0 initial standard deviation of the sampling process

$\Delta\sigma$ increase of sigma for each iteration in the sampling process

τ stopping condition for the sampling process, maximum of the state space covered in percent

θ_{c_p} concept classifier for a concept c in a parameter p

θ_p vector of concept strengths v for a parameter p

Abreviations

HRI Human-Robot Interaction

MDP Markov Decision Process

PDF Probability Density Function

SVM Support Vector Machine

UI User Interface

1 Introduction

Recently advances in sensor technology, machine learning and robotics have widened the operational area of robots from being industrial workhorses for repetitive tasks to more complex applications that require robots to be flexible, adapt to new situations and learn new skills after deployment. This is further emphasized by the growing media coverage that introduces robots to the general public. The trend to incorporate robots in everyday life will lead to an increased number of interactions between robots and humans. This development raises the need for new methods to give robots the tools they need to collaborate and interact with humans. The requirements for such methods are to make the robot's functions, intentions and actions transparent to the human user. Previous research in the field of Human-Robot Interaction (*HRI*) has shown that by helping the user understand these factors about the robot, the human user can form a better mental model that represents the knowledge and expectations about the robot [9]. This makes the robot more predictable and helps the user estimate how much they can rely on the capabilites of the robot. Being able to predict the behavior of the robot makes it more trustworthy and will ultimately support the adoption of robots by the general audience [1, 9, 22, 29].

Next to the social aspects, transparency can increase the efficiency and quality of interactions between humans and robots. In previous research the positive effects of transparency on the performance in various scenarios have been confirmed. These scenarios involve: improving a robot's learning performance in teaching scenarios [5, 6, 7], improving the efficiency of a mixed human-robot team [27] and enabling end user debugging [15].

However, the field of transparency in *HRI* is still new and most research has focused on determining why transparency is important and how it affects interactions between humans and robots. New mechanisms are needed that continue this research and apply these findings to existing systems.

One method how a robot can make its behavior transparent is by explaining its actions to the user. In existing research, explanaitons have mostly been discussed with regard to how users value different kinds of explanations [19, 20]. Further research has focused on which kinds of explanations influence transparency and how. Little research has been done in the area of robot policy explanation [11, 14]. A policy in this context describes the rules on which the robot is making decisions [18]. The existing research either focuses on explaining a policy globally or uses predefined explanations. Thus, the focus of this thesis is to develop a method to generate local explanations that describe specific actions of a robot.

In this thesis we will discuss the effects of transparency in *HRI* and present a framework for local robot policy explanation in a continuous state space. We assume a continuous and deterministic policy. The state space is constituted by the parameters that influence the robot's discrete actions. The goal is to explain why and based on which parameters the robot made a decision and as a result improve the predictability and intelligibility of the system. The developed approach consists

of two steps. First we learn concept classifiers from experts in the fields of robotics and machine learning. These classifiers define concepts as natural language labels over regions in the state space and are used to describe these regions in natural language. Second, we use the concept classifiers to generate explanations for robot actions. As part of this approach we also determine to which degree each parameter is influencing the robot's current actions and adjust the explanations accordingly. The developed framework was tested during a pilot user study in which subjects had to interact with a robot in order to debug its policy using the developed explanation generation framework. Based on the results of our study we got a perspective of how the explanations help users to understand robot behavior and debug erroneous robot behavior. Furthermore we discuss the results and the expert's feedback for our explanation generation framework with regard to a larger future study involving non-experts.

The structure of the thesis is as follows: Chapter 2 gives an overview over the current state of the research in the field of robot transparency in *HRI*. We discuss existing definitions, previous research and point out challenges and benefits of transparency. Chapter 3 introduces the methods and tools used to develop the explanation generation framework. In Chapter 4 we present the structure and design of the user study and discuss the results in Chapter 5. Lastly, Chapter 6 summarizes the work and discusses the possibilities for further research in explanation generation for robot transparency.

2 Transparency in Human-Robot Interaction

This chapter will introduce transparency in Human Robot Interaction (HRI). Transparency in this area has just recently moved in the focus of HRI research [29]. In a survey focused on interactions between robots and human teachers Vollmer and Schillingmann [29] defined Transparency as "Mechanisms in robots that convey the current state of the technical system". While this description is a good summary of the general concept, this chapter will discuss different perspectives on transparency in HRI and compare existing definitions. Furthermore, we outline how transparency in interactions between humans and robots influences not only the course of the interaction itself, but also the way humans perceive robots in their environment. We will present how transparency mechanisms can be applied to explain a robot's decision making and what are the resulting benefits.

2.1 Human Mental Models of Robot Functionalities

The general consesus regarding interactions between humans is that communication leads to better understanding of each others goals and improves teamwork capabilites. Research [26, 24, 12] has shown that humans apply the same principles also when interacting with robots and that humans view agents that show intentions to be a character or creature. This effect is partly a result of the human tendency to anthropomorphize robots by projecting the human idea of social interaction and communication to robots [12], especially when their appearance is suggesting humanoid shapes or behaviors. According to Fink [12], anthropomorphizing can make a non-human entity more familiar, explainable or predictable by attributing familiar properties to it. The human tendency to interact with a robot similar to another human is therefore an important factor to consider when designing HRI applications.

The assumptions humans make about the functions and purpose of robots are often referred to as their *mental model*. DeGreef and Belpaeme [9] describe mental models as a cognitive process in which a person forms expectations about beliefs and goals of another agent. Kulesza et al. [17] further expand the concept of mental models and define two categories of mental models. *Functional Models* imply "that the user understands how to use something but not how it works". *Structural Models* on the other hand provide understanding of both aspects. In this paper and earlier research by Kulesza et al. [16] the benefits of precise mental models have been investigated and confirmed. Their research and how explanations influence the mental model will be discussed in detail in Section 2.3.2.

A concrete example for a transparency application is the experiment in DeGreef and Belpaeme's paper on social robots [9]. In their experiment the performance of a social robot, which was taking social cues into account, and a second robot without social features were compared. The experiment consisted of a human teaching categories to the robot. The results showed that humans are following strategies based on the mental models they have when interacting with a robot and that the teaching performance can be influenced by the robot applying social cues. They

concluded that, in a teaching scenario, users choose their teaching examples based on their mental model of the robot.

Another study by Talamadupula et al. [27] showed that similar to a human, a robot can benefit from a precise mental model as well. Robots can use a mental model to estimate the state of humans and predict needs and actions accordingly and thus improve performance of a human-robot team. In their experiment a rescue team consisting of human commanders and robot assistants performed a rescue operation and the robot tried to predict which medical kits to fetch for the commander based on the commanders location. The results obtained by real world experiments and simulation showed that mental modeling capabilities improved the robot's prediction precision.

Modeling capabilities of humans and robots are beneficial to communicate a precise representation of the capabilities and the state of all agents partaking in an interaction.

2.2 Transparency

With the examples mentioned above, we established the importance of a precise mental model that agents have of each other. This requires each agent to provide transparent information about themselves to other agents participating in an interaction. Transparency, as defined by Vollmer [29], is a powerful tool to shape the formation of mental models. Transparency is providing insight into the internal functions of an intelligent agent such as: [28]

1. Exposing decision making
2. Explaining unexpected behavior
3. Reporting reliability to a human collaborator (or another agent)

In the article "Why is my robot behaving like that? Designing transparency for real time inspection of autonomous robots" [28], Theodorou et al. discuss the importance of transparency and define transparency as an "always on mechanism able to report a system's behavior, reliability, senses and goals". Furthermore, they point out that by applying transparency mechanisms, complex intelligent systems can be demystified and made more accessible for the public audience. From an ethical point of view, the Engineering and Physical Sciences Research Council (EPSRC) [3] agreed on ethical design principles for intelligent robots. Their principle of transparency states that robots should not be designed in order to cover their artificial nature and instead focus on making their machine nature transparent to human users. A more detailed summary of this aspect is presented in [28].

The growing integration of robots and intelligent systems into every day life will increase the importance of the social and ethical aspects of HRI. In order to guarantee widespread adoption, users need to be comfortable with collaborating with robots and trust their artificial team members.

Trust Being a core component of social interactions, trust is a result of having a good mental model of another agent that makes precise predictions possible and increases the understanding of other agents decision making. If humans are able to predict the reliability of a system, they can better estimate to which extent they can trust it to do what they expect. Thus the importance of trust lies in the effect it has on the way humans will rely on the information provided by a robot or displayed by its actions. High levels of trust lead to high acceptance and willingness to collaborate. Distrust can have negative effects like over reliance or on the other hand the abandonment of the robot [13].

An extensive study by Lim et al. [20] has shown that knowing the functionality of an intelligent system creates trust and acceptance. They used different generated explanations of a system's operation log and presented them to novice users of a system that tracks human body functions using sensors and makes decision based on these parameters. The results indicated that explaining decisions to the users increased the general trust as well as the satisfaction and the acceptance of the system.

In his paper "Being Transparent about Transparency: A Model for Human-Robot Interaction" [22], Lyons mentions trust as a key aspect of human-robot interaction. Understanding the trust humans put into their artificial collaborators is important for their adoption in the general public. Furthermore Lyons suggests that transparency does help to create trust, making interactions between humans and their robotic teammates more effective. Lyons suggests two categories of transparency: *Robot-of-Human Transparency* and *Robot-to-Human Transparency*. These categories can be further divided into smaller concepts. Figure 1 shows a schematics of these concepts.

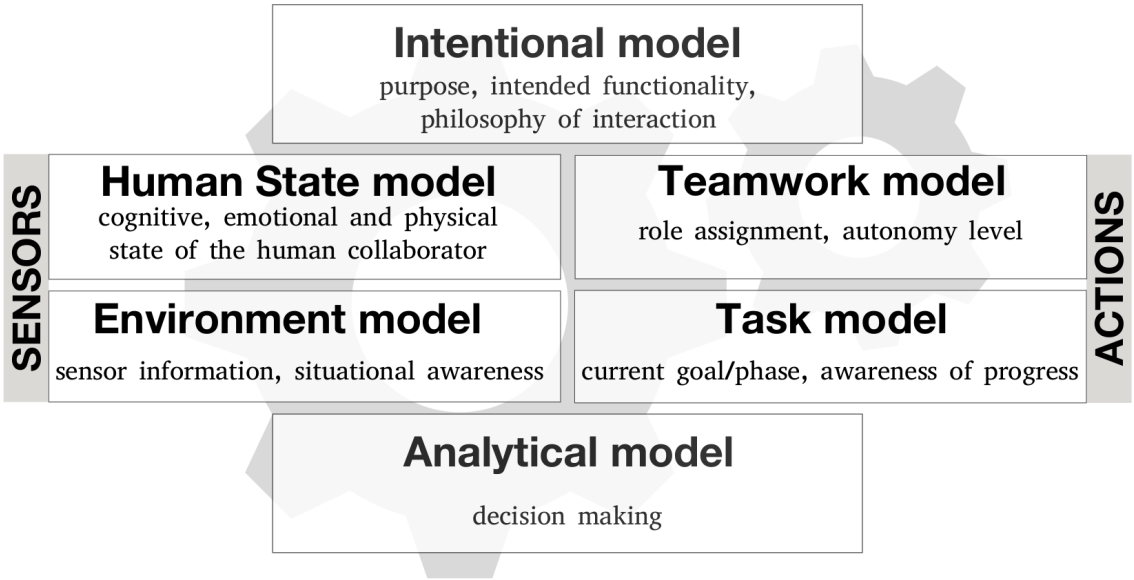


Figure 1: Schematics of Lyons Transparency Classification [22].

2.2.1 Robot-of-Human Transparency

The first sub category, *Robot-of-Human Transparency*, describes the information that a robot has about the state of the human and its environment. Examples for applications utilizing robot-of-human factors are assistants in the automotive and aviation sector. These assistants are designed to supply the driver with information about his state and intervene to guarantee safety. Another factor mentioned by Lyons is concerning the cooperation or teamwork between robots and humans. In order to function as an efficient team the robot has to know about its role in the interaction and communicate it to the human. This allows the human to better predict the actions of the robot. The resulting mental model creates more trust in the robot. Furthermore a precise internal model of the human state can help the robot not to overburden the human and sense physical exhaustion [22].

2.2.2 Robot-to-Human Transparency

The goal of this thesis is to make the internal decision making of the robot transparent to a human user. Lyons includes this concept into *Robot-to-Human* factors. Components of this description are the *Intentional Model*, the *Task Model*, the *Analytical Model* and the *Environment Model*. Each of them describing a different category of information that the robot can provide in order to make certain aspects about its functions transparent. This is furthermore reflected by the definition of transparency in [28] as "a mechanism to expose the decision making of a robot". This definition agrees with Lyons Robot-to-Human transparency concept.

The **Intentional Model** describes the purpose and therefore to a certain degree the capabilities and functions of the robot. According to Lyons, the intentional model should answer the questions "Why the robot was created and how the robot seeks to perform these actions". This effect can be achieved by using physical appearance or providing specific information about why the robot was created. The examples mentioned in Lyons paper are cleaning robots looking like maids and administrative assistants looking like a secretary. Being transparent about the intention of an intelligent system can be challenging if we also consider systems that can not rely on physical appearance, like for example automotive and aviation assistants. Lyons writes that fully understanding the intent of a system helps its users to put the actions of the robot in the proper strategic context, which is especially important if we take the varying knowledge users can have about the robot into account.

The **Environment Model** describes knowledge of the environment in which the interaction is taking place. Communicating this knowledge can help improving the user's situation awareness and help to better understand the influences of the interaction on the environment. This is especially important for mobile robots like service robots and robots in complex outdoor scenarios.

The **Task Model** is closely tied to the intentional model and focuses on providing information about the robot's capabilities in executing a certain task. Lyons describes the task model as "understanding of the current task, information relating to the robot's goals at a given time, the robot's progress in relation to those goals,

information signifying an awareness of the robot's capabilities and awareness of errors". Being transparent about its own capabilities is a key factor for the robot. It can help the human to gain more trust in the robot by providing the human with enough information to predict the robot behavior and failure in certain tasks.

The **Analytical Model** describes the process of communicating the analytical principles or the decision making of a robot. Understanding the decision making can help the human to better understand and predict a robots behavior under uncertainty. Lyons summarizes the analytical model as "a knowledge-based component where the users need to understand the analytical structure of the robot's decision process".

The models introduced by Lyons describe factors that influence the mental model that a human has about a robot. Their main purpose is to make the interaction as natural as possible, provide information and thus create trust in the robot. In this thesis we focus on explaining robot decision making which falls into the category of transparency in the context of the analytical model. Before discussing how we achieve this goal using explanations, we will survey literature on the benefits and challenges of applying transparency mechanisms to intelligent robots.

2.2.3 Benefits and Challenges of Robot Transparency

In this section, we discuss how transparency can improve human-robot interaction. Furthermore, we discuss current challenges when applying transparency to robots.

We will now evaluate at what can be gained from providing transparency mechanisms to robots. As discussed earlier one benefit of transparent and intelligible systems is the increased trust that humans put into them. Studies have shown that trust is usually increased when transparency mechanisms are applied [28, 20, 30]. Next to these benefits, trust leads to increased adoption and acceptance by the general public by making complex agents more intelligible [28]. Especially with the growing embedding of service robots and similar intelligent assistants into our everyday life it is important to consider the social aspect of this development. Theodorou et al. [28] provide the example of service robots for elderly people. They state that trust is an important factor for people with little to no knowledge about intelligent systems and distrust may lead to denying interaction altogether. While increasing trust is a valuable benefit of transparency it is important not to raise wrong expectations and too much trust in the capabilities of a system. Unnoticed over reliance can endanger humans especially in safety critical applications [28].

In the specific case of learning robots, transparency also improves the learning performance as well as quality of interaction. This scenario typically involves a robot learning new tasks from a human. Performance in this context refers to factors that improve learning rate and reduces the time and data required to learn. Cakmak et al. [7] showed that a robot guiding the humans teaching based on existing algorithms and heuristics can increase the learning rate of the robot. In four experiments they compared different types of teaching guidance and evaluated the accuracy of the learner throughout the teaching. The experiments involve a human teaching a robot

about a concept by binary classifying samples. Furthermore, they tracked the number of user interactions during the experiments and the overall duration of the session. The main conclusions are:

1. Without teaching guidance the teaching strategy is rarely optimal
2. The average accuracy of the classifier at the end of a session is higher when the teaching is guided
3. Human teachers are more efficient when they are provided with the instructions on how the robot learns

Further research by Cakmak et al. [5, 6, 7] presents similar results: In scenarios with a human teacher, transparency improves the learning speed and accuracy by giving the human a better understanding of the robot's learning strategy.

By improving the quality of interaction and the learning performance, robots can acquire new skills faster by actively learning from humans. This increases flexibility of when deploying a robot into a new environment [25].

2.3 Transparency through Explanations

As we have established so far in this chapter, transparency improves the mental model humans have of a robot and therefore increase the trust a human has in a robot. The resulting benefits discussed in Section 2.2.3 are a result of the increased explainability and predictability of the robot's behavior. In this section we look at how transparency principles can be applied by explaining the robot's decision process to the user in order to improve intelligibility. Intelligibility as defined by Bellotti and Edwards [2] means that intelligent agents must be able to "represent to their users what they know, how they know it, and what they are doing about it".

2.3.1 Using Explanations to Improve System Intelligibility

Intelligibility with regard to intelligent agents was introduced in [20] and further discussed in [19] as the process of making the rules and underlying machine learning models of a system transparent to a human user. The focus of these papers is on explaining the system's reasoning and determining what kinds of explanations are most beneficial to achieve this goal. The research discussed in this section was done in the scope of Human-Computer Interaction (HCI) using Context-Aware intelligent systems, but the same principles also apply to HRI.

To categorize answers to system related behaviors that have different influences on the intelligibility Lim et al. [20, 19] introduced the following types of questions called *Intelligibility Questions*:

- What did the system do?
- Why did the system do X?
- Why did the system not do X?

- What would the system do if Y happens?
- How can I get the system to do Z, given the current context?

The explanations addressing each of these questions are called *Intelligibility Type Explanations*

In their earlier paper on intelligibility of context-aware intelligent systems [20], Lim et al. show that the explanations answering the different kinds of intelligibility questions influence the understanding of the system and the resulting trust in different ways. In their experiment Lim et al. investigated how automatically generated explanations can improve the systems intelligibility and as a result support the user's understanding. In the experiment a modular web application was developed that allowed to adjust the software so that one application could be used for different experiments. The application created explanations for the underlying input-output system depending on which intelligibility type was studied. To measure which explanations increase the above factors the paper compares the *task performance* as how fast and how precise the tasks were completed by the participants. The other metric was the *user understanding* as a measure for how well the participants understand the system. The experiment showed that explanations answering the "Why?" and "Why not?" intelligibility questions increased both task performance, understanding and as a result also trust in the system. Further analysis showed that "Why?" explanations increase the user understanding more than "Why not?" questions. "How to?" and "What if?" questions showed no significant benefit compared to not having explanations at all.

As a conclusion [20] suggest to use "Why?" explanations over the other types but mentions that specific types of tasks might make the other explanation types viable as well. In their follow up paper [19] Lim et al. conducted two experiments on what information users preferred in explanations and the impact of different intelligibility types on user satisfaction by suggesting questions and explanations. They could confirm the importance of the reportedly more intuitive "Why?" intelligibility type, but found out that users have higher higher expectations regarding the information content of this intelligibility type. This means that even though the answer to "Why?" intelligibility question provided the information, the user might have expected more information than the intelligibility type can provide. As a result the user can be unsatisfied with the interaction. These expectations also rise depending on the user's demand for intelligibility and can introduce a bias towards different types of intelligibility. Compared to the experiment conducted in the earlier study, the web interface complexity was identified as a factor that could introduce confusion and bias the outcome of the study. Another important finding was that in critical situations users want as much information as possible if they know more information is available. Also more complex intelligibility types are more demanded if the user is informed about their availability. In [19] additional intelligibility types are introduced that provides a more detailed schematics with regard to the structure of their experiment.

A more specific application using explanation generation is presented in [15]. In their paper Kulesza et al. introduce *Explanatory Debugging*, a term to describe the method of explaining the learning system's actions. In addition to the explanations

for the user explanatory debugging also includes user feedback to the system in order to increase the speed at which users are able to correct errors of the learning system. Both of these parts are explained by the following two principles: *Explainability* and *Correctability*. Correctability is defined as: "allowing the users to explain corrections back to the learning system" with the goal of "enabling an iterative cycle of explanations between the system and the user". Since the focus of this thesis is on the explanation generation part we will not further discuss this principle here.

2.3.2 The Influence of Explainability on User's Mental Models

Explainability as introduced by Kulesza et al. [15] describes the process of "accurately explaining the learning system's reasons for each prediction to the end user". The goal of explaining a systems actions is to make use of the benefits discussed in the previous section: improving the accuracy of the user's mental model. The paper defines three principles for explanations:

1. Be Iterative
2. Be Sound
3. Be Complete

The first principle, being iterative, refers to the process of refining the user's mental model during an interaction consisting of multiple iterations. It is suggested that the explanations should support this process and consist of easily understandable pieces of information. The second and third principle stress the importance of **truthfully** explaining the **entirety** of the system's components. In addition to the soundness and completeness principles, [15] mentions that it is important to maintain comprehensibility and to keep the user attention in mind. This means finding the right balance of soundness and completeness in order to not overwhelm the user. For example it is possible to simplify complex systems in order to make them more sound at the cost of completeness because the simplification usually has to withold information. On the other hand complex systems can be explained by using simpler examples and thus use a system that is different. This reduces the soundness of the explanation. As example [15] mentioned that a neural network can be explained by using decision trees as a more intelligible example[8]. The paper showed that explanations and feedback in iterative interactions helps users to understand the learning system faster and improve debugging performance. Furthermore the satisfaction of interacting with such a system was found to be higher than for a system without explanation and feedback features.

The soundness and completeness principles have been already introduced in an earlier paper by Kulesza et al. when they investigated the influence of explanations on user's mental models [17]. In this work, mental models are described as internal representations of how an agent works, focusing on how a human uses such a model to debug the agent. As mentioned in Section 2.1, [17] divides mental models into two types, *Functional Models* and *Structural Models*. Structural models were found

to provide more complete information about the functions of the agent and therefore prevent erroneous mental models. In their papers Kulesza et Al. also refer to the intellgibility types by Lim et al. [19, 20] introduced in Section 2.3.1. In [17] it is pointed out that it is still a subject of research how much information an explanation should contain. However in their later work [15], it is mentioned that complete explanations usually contain more intelligibility types. This shows that more information provided by explanations leads to increasing completeness. As mentioned above, finding the right balance between soundness and completeness is important but so far it seems that there is no clear consensus on how much information should be provided to the user and in which situations.

So far we established that careful use of explanations improves the transparency of a system and lead to more precise mental models. These mental models lead to increased user trust, performance and to more rapid acceptance and adoption. After establishing why explanations are useful and how they provide transparency, we want to now focus on generating explanations for a more specific scenario: the explanation of a robot's policy.

2.3.3 Robot Policy Explanation

The research presented in Section 2.3.2 has mostly focused on the influence of explanations as well as how the explanations should look like. Our work focuses on how we can generate such explanations automatically for a robot's policy. In his book "Planning Algorithms" [18], LaValle defines policy as a plan or control law for an agent. In the scope of this thesis the agent is a robot with a policy that we want to explain to a human user. The policy is the underlying logic that leads the robot to take a certain action. In this chapter we will discuss the existing research regarding policy explanation.

In their paper Elizalde et al. [10] investigated how explanations improved the performance of operators that were trained with an explanation generation mechanism. The experiment showed that operators trained with these mechanisms performed better than a control group without. However, the explanations in this earlier experiment were generated by experts in the field and the generation of expert explanations for every scenario is not flexible and very laborious. Thus automatically generated explanations are a current subject of research [11].

In their later paper on policy explanation Elizalde et al. [11] focus on automated explanation generation. In an experiment with a recommender system for a simulated steam generator Elizalde et al. applied the knowledge form their previous research and investigated the use of automatically generated explanations. The explanations were generated using a Markov Decision Process (MDP) and described what actions to take based on the parameter of the generator. A MDP is a discrete time stochastic control process and often used in robotics for decision making in scenarios that involve random outcomes that can be influenced by the robot [18]. In a MDP each discrete step is described by the mapping of a state S and action A to a new state S' [11, 18]. By analyzing the explanations provided by human experts they found out

that explanations were based on the perceived most important variable. In this regard the most important variable refers to the variable that has the highest influence on the choice of the action. As a result of this they introduced heuristics for determining relevant variables or parameters in a MDP to generate more precise explanations. We use this assumption as an important factor when generating explanations because we expect explanations that are using the same parameter priorities as humans to be more human like and predictable. In that sense we are applying Robot-of-Human factors introduced in Section 2.2.1 and try to converge to the human mental model of what an explanation has to look like. In [11] two heuristics for finding the most relevant parameter are introduced: the *Utility-based heuristic* is based on the impact that one variable has in the given state. To determine the impact each parameter is varied while the others stay fixed and the resulting utility is evaluated. The parameter that leads to the highest change in utility is selected as the most relevant parameter. The *Policy-based heuristic* is based on the impact of a parameter on the potential change of the optimal action. Similar to the utility-based heuristic one parameter is varied and the others kept fixed. Then the parameter that leads to the highest potential change in the policy is selected as the most relevant. The experiment conducted in the paper showed that the selected most relevant parameter generally agreed with the expert selected parameters. Furthermore the utility-based heuristic provided more accurate results.

This work on the generation of explanations [10, 11] has been continued by Wang et Al. [30] who investigated the influence of explanations on trust in a mixed human-robot team. Wang et Al. introduced uncertainty to the robots representation of the environment by using a *Partially Observable Markov Decision Process* (POMDP). The explanations were generated by translating parts of the robot's POMDP model into natural language using templates. In an experiment human participants were collaborating with a simulated robot. The trust that the subjects put into the robot was measured by comparing the perceived trust levels of the participants. The performance was evaluated by measuring the participants understanding of the robot's decision making. Furthermore various other established performance tests were conducted [30]. This paper presents a solution for how to measure the influences of explanations on the trust and performance. It will serve as inspiration for the analysis of the experiment conducted in this thesis.

While the research by Elizalde et al. is focused on explaining a single event in the decision making, Hayes and Shah investigated methods to generalized explanations and make the entire control logic transparent. In their paper on autonomous robot policy explanation [14], Hayes and Shah introduced a system that enables a robot to autonomously generate a description for its policy. In their approach, code annotations are used to learn a model of the robot's policy from observing the controller. During the execution of the control logic the code annotations are used to detect functions calls. For each function call the state, action and target state are recorded and a MDP is constructed. The MDP encodes the information which states lead to which actions and serves as the framework to generate explanations. The MDP encoded policy is used to track which states lead to which action and find

a concept that best summarizes these states into an explanation.

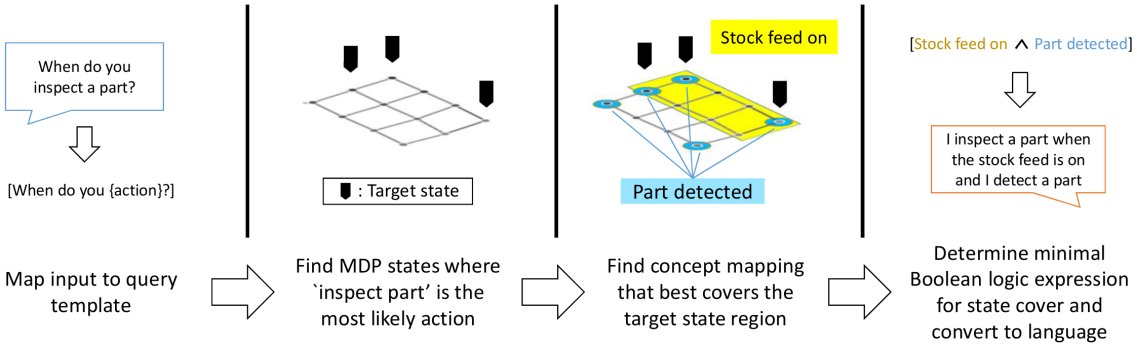


Figure 2: Schematics of the policy explanation process by Hayes and Shah [14].

Figure 2 shows the policy explanation process in [14]. In the first step the user can query the system to explain the policy depending on different query types. It is possible to query the system with the following templates:

- When [do|will] you [action]?
- Why [didn't|aren't] you [action]?
- What will you do when [state region description]?

Although it is not mentioned in the paper, these templates seem to follow the intelligibility types introduced in Section 2.3.1. However instead of focusing on one specific question, they refer to the whole policy. This allows the robot to explain its general policy but is limited in explaining a certain action locally.

After the query, the system will determine all states that are likely to be the result of the action from the query. The determined region of states is mapped to concepts and the resulting boolean expression is simplified by finding its smallest logical representation. This is necessary because the explanations are more complex the larger the state space is. In their paper Hayes and Shah used Boolean logic minimization after Quine-McClusky [23].

The explanation framework was tested in three different scenarios with 3 different robot controllers. For all three case studies the robot successfully generated explanations that resemble expert explanations. The paper does not provide detailed discussion about the quality or benefits of these query types, but in the result discussion the explanations generated using the "When do you [action]?" query template are compared to expert policy explanations. It can be seen that the generated explanations resemble the experts explanations to a high degree.

However we can see that the resulting explanations are statements about one certain state that can be true or false. One explanation from the table in [14] is : "I move north when I am south of a delivery area and have the part. I move east when I am west of a delivery area and have the part and not near a human zone". As a result of that, the discrete nature of the MDP encoded policy limits the explanations

to sequences of boolean statements. Furthermore the controller has to be observed and the MDP generated before the framework is ready to use. This leads to limited flexibility and is impossible if the variables that lead to decisions are not binary. For example with the method introduced in this paper it is not possible to describe concepts over a continuous state space or a variable that consists of three concepts like "high, middle, low".

3 Local Policy Explanation with Expert-Learned Concept Classifiers

After establishing how we can make a robot's decision making more transparent, we now introduce our automatic explanation generation framework.

3.1 Generating Explanations

The explanations we provide to the users are generated using natural language templates. Our template provides an answer to the intelligibility question *"Why did the system do X?"*. This template was chosen because it is, compared to other intelligibility types, the simplest one while still providing information about the current action [20]. To translate regions in the state space into natural language we use concepts as a label for an area in the state space. Expert learned concept classifiers are used to explain the action the robot took following the natural language template: *"I did [Action] because [Parameter] was [Concept]"*. The explanation templates are based on the intelligibility types by Lim et al. [20, 19]. Furthermore the focus of this thesis is to investigate the generation of local explanations and the influence of more complex explanations is left for further research.

3.2 Internal Representation of the Robot's Policy

The representation of the policy in the state space will be the foundation on which we base our method for explanation generation. In previous research discrete representations of the robot's control logic were used. Our method uses a continuous representation of each variable or dimension that the robot uses to make its decisions. We will refer to these dimensions as *Parameters* P from here on. Previous research has mostly focused on the influence that questions have by using predefined explanations or explanations learned from experts [10, 11]. Furthermore the current state of the research has only explored global policy explanation in discrete space [14]. The benefits of automatically generated explanations in continuous state space are that we are not limited to a fixed set of explanations or a discrete state space. In addition, local explanations provide the ability to iteratively debug and learn about the current state of the robot[11, 14]. Our approaches continues this existing research and brings local explanation generation to continuous state spaces.

A state s describes the current state of the robot in the state space S. a is an action that a robot can take to transform its current state s into a desired goal state s'. Thus we can define a policy π as a rule that defines the action of the robot for any state in the state space [18] as

$$\pi(s) = a, \tag{1}$$

where $s \in S$ and $a \in A$ with S being the state space and

$$S = \{p_1 \times p_2 \times ... \times p_N\}, \tag{2}$$

as a set of individual dimension of S. $p_1, p_2, ..., p_N \in S$ are the continuous parameters that constitute the state space space S of N dimensions.

The discrete action set A of M actions is defined as

$$A = \{a_1, a_2, ..., a_M\}. \tag{3}$$

During this chapter we will use an example to aid the understanding of the explanation generation framework. We use a simulated vacuum cleaning robot that has to balance its battery level and the dirtiness of a room in order to efficiently clean. The dirtiness and the battery level constitute a 2-dimensional continuous state space. In our example we used a support vector machine (SVM) as policy for the simulated vacuum cleaning robot. The resulting policy in the 2D state space is displayed in Figure 3.

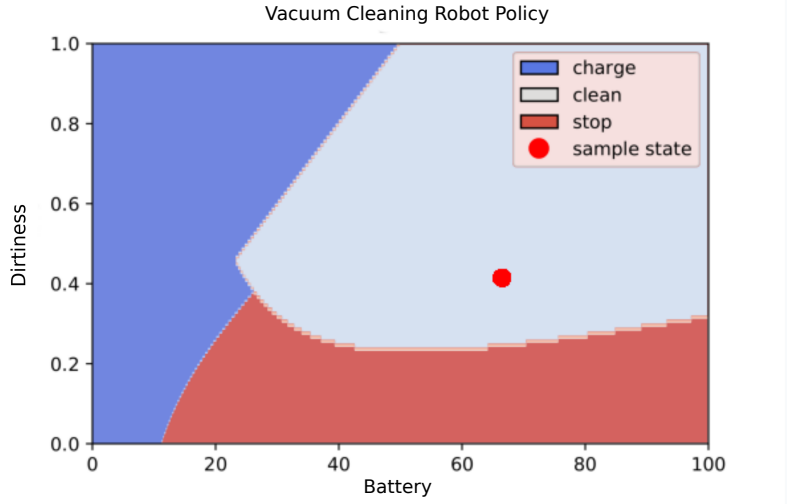


Figure 3: Example continuous 2-D state space with a support vector machine as policy.

3.3 Learning Concept Classifiers From Experts

In order to generate natural language explanations, we attach labels to the state space. These concepts provide a layer of abstraction so that information about the state space can be understood easier than numerical representations. We define a set of concepts C_p for a parameter p as

$$C_p = \{c_1, c_2, .., c_J\}, \tag{4}$$

where $c_1, c_2, ..., c_J$ are the J concepts in parameter p. To label the state space with concepts we use classifiers that attaches a natural language label to a region of the state space.

Examples for concepts in context of the vacuum cleaning robot are: "high battery value" or "low dirtiness" for the battery and dirtiness parameters. The concepts

help to generate intelligible explanations based on an expert's understanding of the complex state space.

To determine the concept classifiers, we use randomly generated sample states in the state space and query the expert to determine which concept each parameter of the sample corresponds to. In the example introduced in Section 3.2, a sample state consists of a value for all P parameters of the state space: the battery charge of the robot and the dirtiness of the room. Labels for these samples are collected and N classifiers are trained where $N = J \times P$ is the number of all concepts c in all p. In our example we used a normal distribution to model a concept. In Figure 3, an arbitrary sample is marked as a red dot at 20% battery level and a dirtiness of 0.6. The provided label would be: "Battery Low" and "Dirtiness Medium".

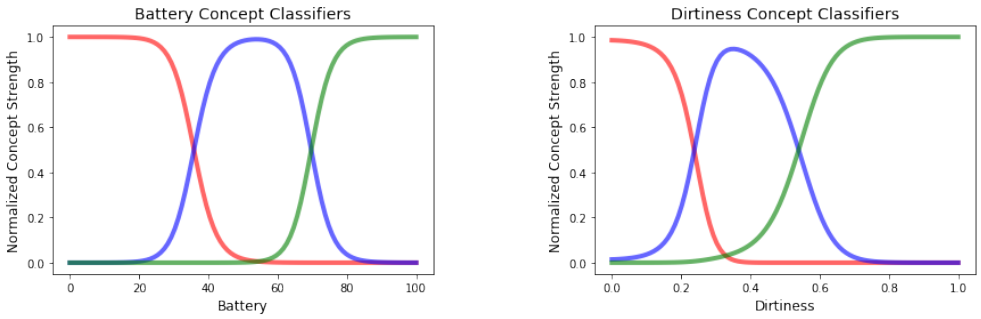
For the vacuum cleaning robot example, we used an expert to provide the concept labels for the battery and dirtiness parameters. The expert was familiar with the robot behavior and the goal of the experiment in order to ensure good concepts.

The first decision one has to make when generating concept classifiers is the number of different concepts for each parameter. Too many concepts lead to reduced intelligibility of the explanations because the user might be overwhelmed by the high number of concept nuances in the explanation. Too few concepts might not provide enough detail to explain complex policies with enough details. In order to chose the number of concepts, it can help to take the policy of the robot into account. Depending on how precise we want to model the state space and how many actions have to be explained, the number of concepts can vary. For our example, we can take Figure 3 for reference: a good trade off between simplification and precision can be two to four concepts for each parameter.

After gathering the samples to fit the normal distributions, we have C_p probability density functions PDF, one for each concept c in each parameter p. A normalized concept classifier θ is defined as:

$$\theta_{c_p}(s) = \frac{PDF_c(s)}{\sum_{c \in C_p} PDF_c(s)} \tag{5}$$

The resulting normalized concept classifiers for our example are shown in Figure 4.



(a) Concept classifier for the battery parameter

(b) Concept classifier for the dirtiness parameter

Figure 4: Normalized concept classifiers for a vacuum cleaning robot example

One normalized classifier θ_{c_p} for a concept c in a parameter p can be interpreted as following: the x-axis represents p and the y-axis shows the concept strength v. Each of the curves shows the concepts strength of one concept c. The concept strength for $x \in p$ can be calculated as follows:

$$v_c = \theta_{c_p}(x) \tag{6}$$

Thus θ_p as the classifier for a parameter can be defined as a vector of concept strengths returned by the individual classifiers θ_{c_p}.

$$\begin{bmatrix} v_1 \\ v_2 \\ \dots \\ v_J \end{bmatrix} = \theta_p(s) \tag{7}$$

For example if we assume that the battery level is at 25%, the classifiers from Figure 4 would return the following values for:

Classifier $C_{concept}$	Concept Strength v_c
$C_{BatteryLow}(25)$	0.97
$C_{BatteryMiddle}(25)$	0.03
$C_{BatteryHigh}(25)$	0.0000

From the above table we can see that the example battery value of 25% can be classified as "Low Battery".

3.4 Explanation Generation

With the concept classifiers θ_{c_p} learned, we will now focus on how the system selects concepts for explaining the state s_t that leads an action a_t.

The explanation generation process consists of three phases. The first step is to sample the state space around the current state s_t in the state space and calculate ratios of samples for each concept. Sampling allows us to apply this approach to a continuous state space. Then the ratios for each parameter are weighted in order to account for the differences in influence that each parameter has on the course of the policy $\pi(s)$ in the vicinity of s_t. After weighting the parameters we can select the most significant concept and generate an explanation using templates.

3.4.1 Explaining Local State Space by Sampling

The first step in generating an explanation for the current state of the system is to sample around it and learn more about the state space in the vicinity of the state. By sampling around the current state, we can learn how the policy in the vicinity of s_t is defined.

Following the vacuum cleaning robot example, a sampling process with two parameters $[p_1, p_2]$ having concepts $[c_{1,1}, c_{1,2}, c_{1,3}]$ and $[c_{2,1}, c_{2,2}]$ is shown in Figure 5. The red lines represent the boundaries that separate concepts from each other. In

this example, the state s_t that is to be explained is represented by a red dot. The first step when initializing the explanation generation with a state s_t is to sample S samples around it. For sampling we use a Gaussian normal distribution $\mathcal{N}(\mu, \sigma)$. The normal distribution allows us to sample around the current state while focusing the sample density on the state space around s_t which we want to explain. With he standard deviation σ we can control the region in the state space covered. In Figure 5, s_t is represented by a red dot. The samples are divided by the action boundary π into two actions a_0 and a_1. This blue line is the representation of π and indicates a border where the actions change from one to another. The samples in action 0 are marked by yellow dots and the samples in action 1 are marked by purple dots. The sampling process starts close to the initial sample with a value for the standard deviation σ. σ is increased until an explanation has been found. The spread of the samples is scaled according to the parameter ranges so that we sample over the state space. We assume that the parameter ranges used for scaling are determined when learning the parameter classifiers from the experts as described in Section 3.3.

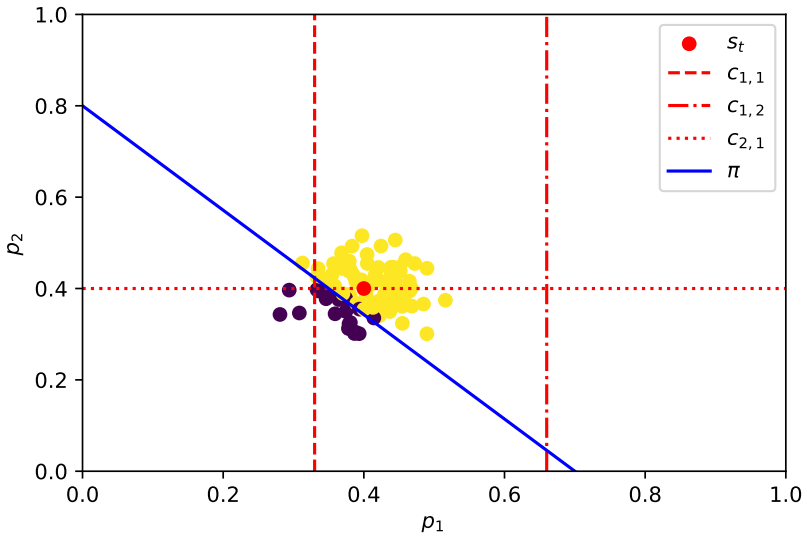


Figure 5: Explanation generation sampling process after 8 steps with $\sigma = 2.5$

Since in the beginning we do not know how local the sampling has to be we start off with a value σ_0 as the initial standard deviation σ and repeat the process for multiple iterations. In our experiment we set $\sigma_0 = 1$. After an iteration the initial standard deviation is increased by a set value $\Delta\sigma$. In our example we set $\Delta\sigma = 0.5$. σ_0 and $\Delta\sigma$ can be chosen based on how local the explanation generation should begin and how fast the sampling space size should increase. This decision also depends on the state space and the policy. If the policy is complex and there are many different actions it is better to sample more locally than for large state space with a simple policy. The standard deviation is increased every iteration by $\Delta\sigma$ until we meet stopping conditions and the explanation is generated. Determining when to stop sampling is a crucial part of the system and will be explained in detail in Section

3.4.3. For now we focus on what happens when a stopping condition is met and the explanation is generated. When the incremental sampling is stopped and the explanation is generated, we calculate a concept ratio r for all samples $s \in S$ in the sample space. First we define γ_p as the concept of a sample s. γ_p is calculated by finding the concept in a parameter p for which the value $v_c(s)$ is the greatest value compared all concepts.

$$\gamma_p(s_p) = \underset{c}{\operatorname{argmax}}(v_c(s_p)) \tag{8}$$

with s_p being the dimension of the sample that refers to the parameter p (see Equation 2).

Then r is calculated as the ratio between n, the value of the samples that have the same action a and concept c to d, the values of the samples that have the same concept. We calculate n and d as

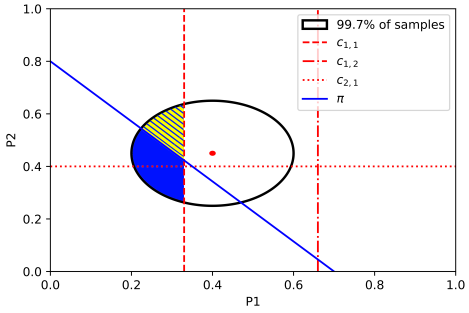
$$n_{c,a} = \sum_{s \in S | \gamma(s)=c \wedge \pi(s)=a} v_c(s), \tag{9}$$

$$d_{c,a} = \sum_{s \in S | \gamma(s)=c} v_c(s), \tag{10}$$

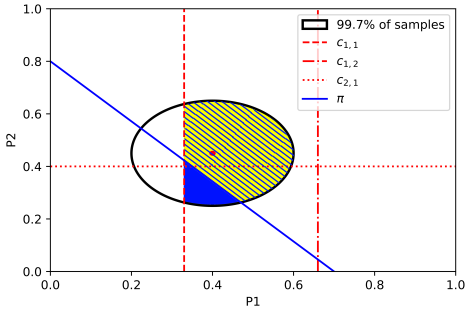
with c_{s^*} being the concept of the current sample s^*. The concept ratio $r_{c,a}$ is thus defined as

$$r_{c,a} = \frac{n_{c,a}}{d_{c,a}}. \tag{11}$$

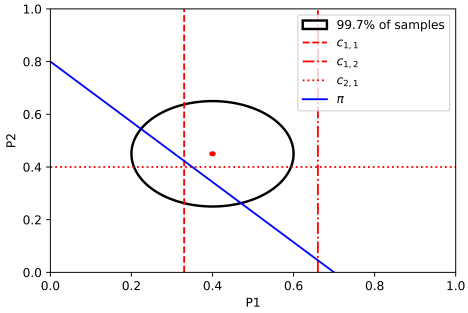
r gives us the influence that each concept has on the mapping $s \rightarrow a$. The larger the ratio, the more important the given concept. In the examples in Figure 6 and 7 the policy π is plotted as a blue line. The circle is a simplified representation of our samples and indicate the interval of 3 σ, in which 99.7% of samples will be. All samples that falls into n are represented by the yellow area and the samples in d are marked as the blue area. The red lines show where in the state space two concepts are equally likely according to the concept classifiers. $c_{1,1}$ to $c_{2,1}$ are the concept boundaries that separate the areas in which a certain concept is strongest.



(a) Concept Ratio for Concept $c_1 \in p_1$, $r_{c_1} = 0.29$

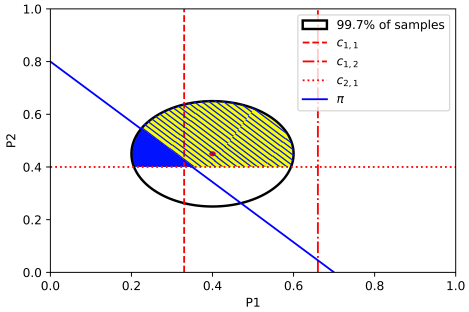


(b) Concept Ratio for Concept $c_2 \in p_1$, $r_{c_2} = 0.78$

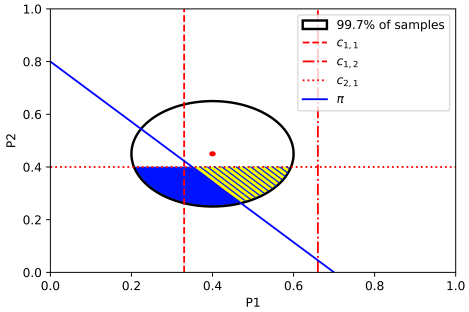


(c) Concept Ratio for Concept $c_3 \in p1$, $r_{c_3} = 0.0$

Figure 6: Example: Concept Ratios for Parameter 1



(a) Concept Ratio for Concept $c_1 \in p_2$, $r_{c_1} = 0.72$



(b) Concept Ratio for Concept $c_2 \in p_2$, $r_{c_2} = 0.39$

Figure 7: Example: Concept Ratios for Parameter 2

When calculating n and d we are calculating v using Equation 6, which is the output of the corresponding concept classifier. v_s describes the strength of the current sample weighted by how much it contributes to the current concept.

Furthermore we are using a scale factor m to account for the different numbers of samples in each concepts. The number of samples in a concept are defined as

$$S_c = \sum_{s^* \ in S | c = c_{s^*}} 1, \tag{12}$$

with c_{s^*} as the concept of the current sample s^* defined as in Equation 8. The scale ratio for a concept c is defined as

$$m_c = \frac{S_c}{S}, \tag{13}$$

with S as the set of all samples. Too few samples can lead to parts of the sample space not being covered. Too many samples can have a negative influence on the computation time. For our example we set $N = 200$ to provide a coverage of the space we want to sample. In Figure 6 and 7 we can see that in our example c_1 in p_1 contains usually fewer samples than c_2 and we have to correct this to weight concepts correctly.

The following algorithm shows the sampling process in detail.

Algorithm 1 Sampling

INPUT: N, s_t, P, π ▷ number of samples, state, action, parameters, policy
OUTPUT: r_c, samples ▷ array of ratios for each concept, samples

samples ← *sample N times from distribution* $\mathcal{N}(s_t, \sigma)$

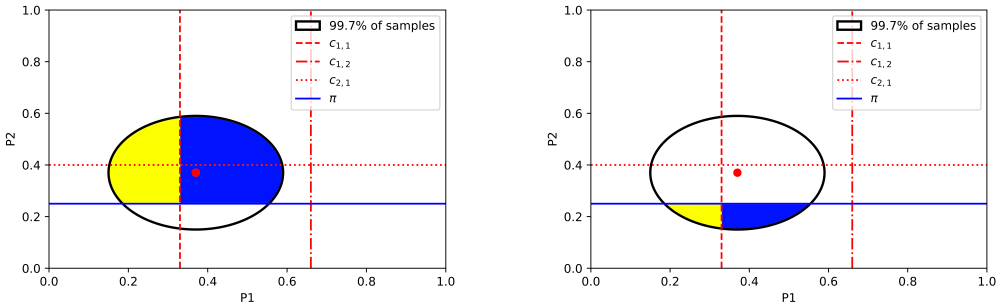
for s_i *in samples* **do** ▷ Iterate through all samples
  for *all parameters p in P* **do**
    for *all concepts c in p* **do** ▷ Calculate n and d for all concepts c in each parameter P
      $v_s \leftarrow \theta_{c_p}(s_i)$ ▷ See Eq. 6
      if $\pi(s_i) == \pi(s_t)$ **then**
        $n_c \leftarrow n_c + v$
      $d_c \leftarrow d_c + v$

for all *all parameters p in P* **do**
  for all *all concepts c in p* **do** ▷ Calculate the ratio for all concepts c in each parameter P
    $m = \frac{S_c}{N}$ ▷ See Eq. 13
    $r_{c_p} = m \times \frac{n_c}{d_c}$

The result of this algorithm is an array of ratios r_{c_p} for each concept c_p in each parameter p. The concept ratios tell us us how significant each concept is for explaining the policy in this region of the state space. For the example shown in the Figures 6 and 7 the ratios for Concept c_2 in Parameter p_1 and c_1 in p_2 are the largest ratios. We can see that this representation describes the concepts that would generate an explanation for the sample.

3.4.2 Weighting the Parameters

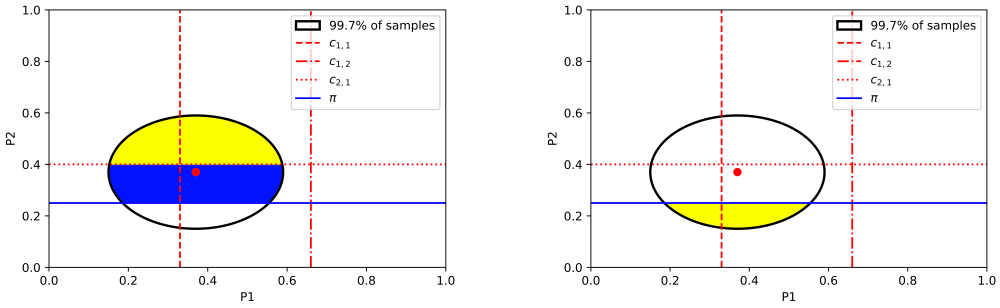
Weighting the parameters is an important step to identify parameters that influence the policy in the vicinity of a sample that we want to explain. As presented in [11], expert explanations about a system usually focus on describing the influence of the most important parameters. We use this approach to improve our local explanations by weighting the concept ratios generated in by Algorithm 1 in Section 3.4.1. This takes into account examples close to the border that separate different actions or with parameters that do not influence the action. Especially the last case can be misleading by generating an explanation using a parameter that is not relevant for the decision making. Looking at Figure 8 and 9 showing the scale ratios for an arbitrary state of the vacuum cleaning robot, we can see that the parameter p_1 has no effect on the action boundary π. The boundary, represented by the blue line, does not change for different values of p_1. In this example the ratios would be higher for c_2 in p_1 than for c_1 in p_2 for being more distinct. Thus the algorithm would select a concept from a parameter that is not influencing the policy at all. The conclusion from this example is, that the smaller the change of the action boundary along a parameter is, the smaller is the significance of this parameter for the explanation. In the case of a linear policy the slope of the policy in the area covered by the sampling would indicate how strong the influence of a parameter is. If the policy boundary is parallel to a parameter this parameter has no influence and if it is perpendicular it has a high influence. For example in Figure 8 and 9 the policy is parallel to parameter 1 on the x-axis and thus there is no change in the policy for different values of p_1. No matter what is the value of p_1, the decision for an action stays the same. This means that this parameter will be scaled to zero. p_2 on the other hand is perpendicular to the policy and and thus the most important concept for the explanation. p_2 decides which action will be taken.



(a) Scale Ratio for Action 0 in Parameter 1

(b) Scale Ratio for Action 1 in Parameter 1

Figure 8: Example: Scale Ratios for Parameter 1



(a) Concept Ratio for Action 0 in Parameter 2

(b) Scale Ratio for Action 1 in Parameter 2

Figure 9: Example: Scale Ratios for Parameter 2

The previous examples are extreme cases for parameters that have no influence at all on the policy. To calculate the magnitude of influence that a certain parameter has, we need to know more about the course of the policy in the vicinity of s_t. For each parameter p and action a, we create a histogram $h_{a,p}$ to determine how many samples are in which concept region. Each concept is thereby a bin in the histogram. In order to determine the change of the policy within a parameter, we compare the histograms. If all histograms h for a parameter p are similar then we know that this parameter has little influence on the policy. Then we use the Jensen-Shannon distance as a measure of how similar the histograms for one parameter are. The resulting histograms generated form Figure 8 and 9 are shown in Figure 10.

The Jensen-Shannon distance is used to measure the similarity of probability distributions and therefore is a good tool for our purpose of comparing distributions of samples [21]. The Jensen-Shannon distance d_p for a two sample distributions P, Q in a parameter p is calculated using the Jensen-Shannon divergence JSD which is based on the Kullback-Leibler divergence D. The Jensen-Shannon divergence for two probability distributions P and Q [21] is defined as

$$JSD(P||Q) = \frac{1}{2}D(P||M) + \frac{1}{2}D(Q||M), \tag{14}$$

with:

$$M = \frac{1}{2}(P + Q). \tag{15}$$

Thus the Jensen-Shannon distance is defined as

$$d_p(P, Q) = \sqrt{JSD(P, Q)}. \tag{16}$$

High similarity, or a low Jensen-Shannon distance, means that the given parameter has no influence on the action choice and thus there is no change between the histograms throughout the different actions. Looking at the example in Figure 8 and 9 we can see that p_1 is not relevant because its Jensen-Shannon distance is zero, which indicates that the action in this parameter does not change. In Figure 10 the

Jensen-Shannon distance between the histograms h_{a_0,p_1} and h_{a_1,p_1} is 0.005. Between h_{a_0,p_2} and h_{a_1,p_2} the Jensen-Shannon distance is 0.15.

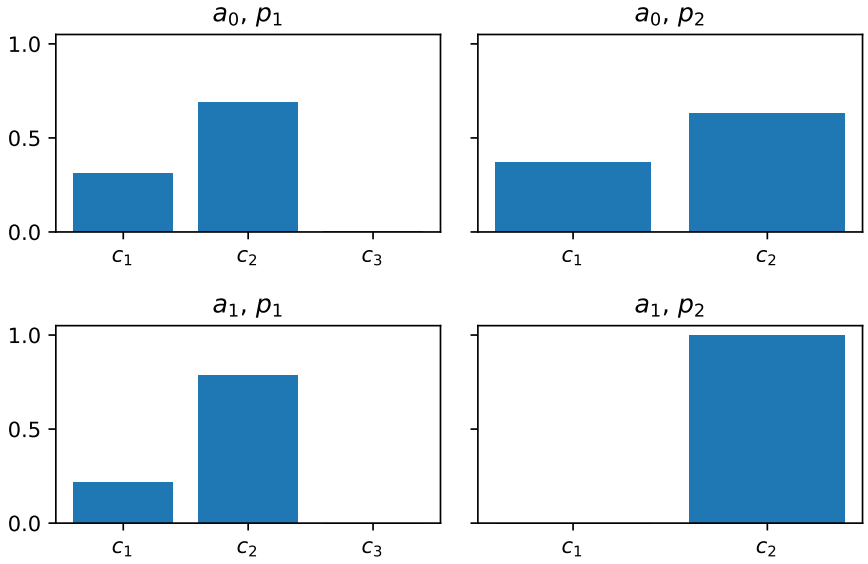


Figure 10: Example: Sample Histograms for each Action and Parameter Combination

As an example, the histogram for p_1 and a_1 is generated by counting the number of samples in each concept of p_1 that are leading to a_1. One concept is one bin h_{a,c,p^*} in the histogram. A bin for a concept c in parameter p^* with action a is defined as

$$h_{a,c,p^*} = \sum_{s \in S | s \in c, \pi(s)=a} 1. \tag{17}$$

Based on these bins we can define a histogram as a set of bins for an action a and a parameter p:

$$h_{a,p} = \{h_{a,c_1,p}, h_{a,c_2,p}, ..., h_{a,c_J,p}\}. \tag{18}$$

We calculate the Jensen-Shannon distance between all histograms using Equation 16 as

$$scale_p = \frac{\sum_{i=0}^{M} \sum_{j=0}^{M} d_p(h_{a_i,p}, h_{a_j,p})}{\binom{M}{2}} \tag{19}$$

and use the binomial coefficient to average them into a scale value $scale_p$ for each parameter p. Averaging allows us to also take more complex states in the state space into account. When for example multiple actions are possible we can still scale the parameters but at the cost of losing information with an increasing number of actions. The parameter scale values are then used to scale the concept ratios for each parameter accordingly by

$$r_c^* = r_c * scale_p. \tag{20}$$

We introduce Algorithm 2 which is executed directly after Algorithm 1 with its output parameters as input. Both algorithms are executed sequentially within one sampling iteration. Then the sampling radius is increased and the next iteration begins. Algorithm 1 is weighting each concept ratio using the Jensen-Shannon distance and the histogram approach introduced above.

Algorithm 2 An algorithm to weigh parameters according to their influence on the policy

INPUT: $s_t, a_t, r_c, samples, \pi, seg$ ▷ state, action, concept ratios and samples from Algorithm 1, policy, number of segments
OUTPUT: r_c^* ▷ Weighted concept ratios used for explanation generation

▷ Count number of samples in each concept for each Action and Parameter Combination
for all parameters p in P **do**
    $hist \leftarrow [\,]$
    for all actions a **do**
        for all concepts c in p **do**
            $hist_{a,p} \leftarrow$ histogram for a and p as calculated in Eq. 18
    $scale_p \leftarrow$ scale value for parameter p as calculated by Eq. 19

▷ Scale each concept ratio with the Jensen-Shannon distances of the parameter
for all parameters p in P **do**
    for all concepts c in p **do**
        $r_c^* \leftarrow r_c * scale_p$

Algorithm 2 takes the concept ratios r_c for each concept as input and weights them according to their influence on the action selection. The returned weighted concept ratio r_c^* is the weighted version of the concept ratios r_c. In a first step the algorithm creates histograms for each parameter and action combination. In our example as seen in Figure 10, this gave us four possible combinations between two actions (a_1, a_2, p_1 and p_2).

During this process the Jensen-Shannon distances for each parameter are calculated and stored. In the second step the Jensen-Shannon distances for each parameter will be multiplied with the concept ratios r_c in the same concept that was calculated during the sampling process in Section 3.4.1. Since higher similarity of histograms means lower Jensen-Shannon distance, the parameter ratios will be scaled accordingly. In our example, the resulting scaled concept ratios for the insignificant parameter p_1 are almost zero (due to the high similarity) while the concepts for the significant parameter p_2 are amplified.

Furthermore this algorithm can be used to find out when the sampling radius is large enough that an action border has been found. If no different action occurs in the sample space, the Jensen-Shannon distances are the same for every parameter and the concept ratios are scaled equally. The result will be that the strongest concept

will be selected for the explanation and the policy, if it exists in the first place, does not have any influence on the weighting. Thus, if the Jensen-Shannon distances are all the same, we can assume that no action boundary has been found yet. This will be applied in the next section to determine when we found the policy in the state space.

With these results we can select the concept with the highest weighted concept ratio r_c^* of them all. This gives us the concept that influenced the action selection in the area the most and translate it into a natural language explanation.

3.4.3 The Sampling Process

A crucial part of generating an explanation for a state in the state space is to decide when to stop the sampling process and generate an explanation. As previously mentioned we are using normally distributed samples around s_t. The sampling process begins with a small value for the standard deviation σ_0, depending on how local we want the explanations to be. We looked at the policy to decide how to chose σ_0. If the policy is not complex, like in our example with 3 actions, we can choose σ_0 larger to cover more area in the state space and find the policy border faster. With the same considerations we then select a value for $\Delta\sigma$, the increase of σ every iteration of the sampling process.

Once σ_0 and $\Delta\sigma$ are selected, both Algorithm 1 and Algorithm 2 are executed iteratively until we meet the conditions to generate an explanation. In the current implementation this condition is set to stop sampling and generate an example if we found an action boundary. This means to sample until the action border was found and define a number of samples that have to be on the other side of the action boundary within a 99% confidence interval. To not sample the whole state space in case we do not find this boundary, we furthermore define a variable τ that indicates how much of the state space can maximally be covered before sampling is stopped and an explanation is generated. We use the value of σ needed to make sure that $\tau = 50\%$ of the state space is within the 99% confidence interval as stopping condition. This approach allows us to define how local the explanation should be. However it is important to note that if no action border was found, the parameters will not be weighted and just use the strongest concept within the sampled area as explanation. These action borders can be found using Algorithm 2 as described in Section 3.4.2. As example on how this can cause problems we look at an example in Figure 11. We can see that if the sample was far away from the action border and we chose small values for σ_0 or $\Delta\sigma$, the explanation could select a concept from parameter p_1 which is not contributing to the action decision and therefore not a valid explanation.

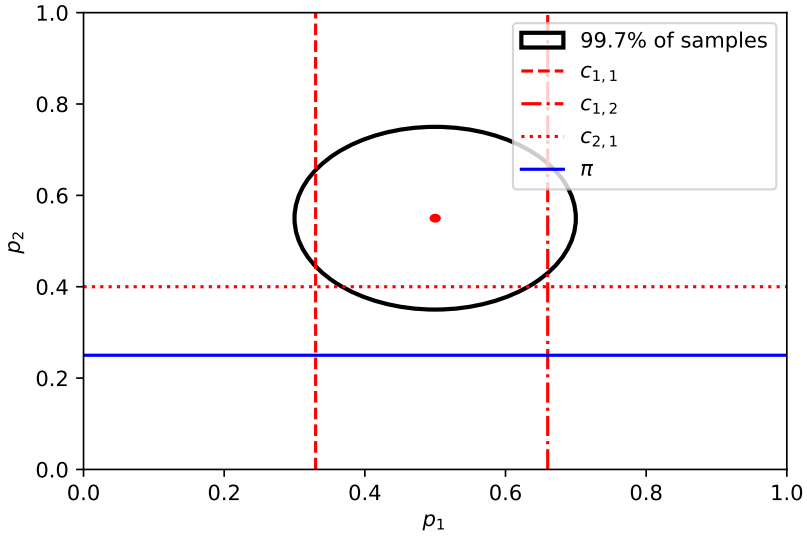


Figure 11: Example of p_1 having no influence on the policy boundary

To solve this, one can instead always sample until the next policy boundary was found. This solution would solve the above problem but can lead to explaining the policy too global in these extreme cases. Furthermore if there is no policy within a reasonable distance in the state space, the sampling can take very long or not find the policy at all. For this approach we therefore have to know that there is a policy in the state space and that the state space is limited. Further research is required to investigate how this approach can be optimized for unlimited state spaces and how τ influences the locality and therefore the quality of the explanations.

3.5 Classifying Complex Concepts

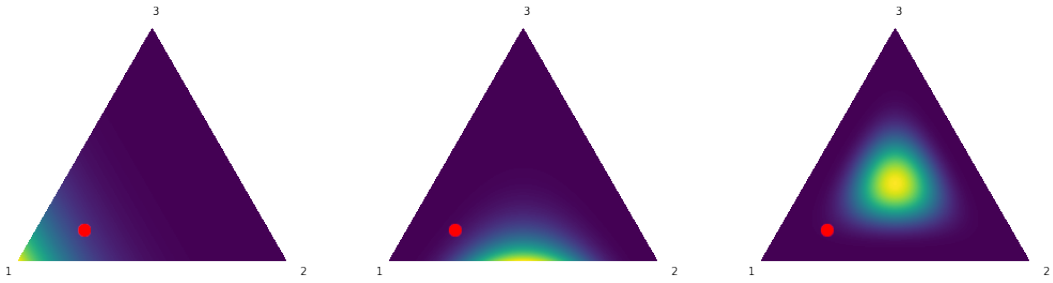
So far we discussed how to create concept classifiers for simple linear concepts. In this section we present how this approach works for more complex concepts like the probability vector output of a classifier. In this section we use an arbitrary probability vector as output of a classifier with 3 classes. p is an example for such a probability vector.

$$p = [0.17, 0.90, 0.22]$$

The goal is to simplify the complex concept so that they are more intuitive and understandable for human user. Examples for such simplified concepts v_1, v_2, v_3 in our example could be:

- Sure about 1 class
- Either of 2 classes
- Unsure about all classes

To generalize the concepts the vector is sorted by probability in descending order. The sorting allows us to apply the same concepts independent of the classes since their order does not matter for the general concepts we chose. For example if we are sure about one class it does not matter if after sorting this is class 1 or a class N. After ordering the vector, we use a **Dirichlet Distribution** as a classifier to model concepts in a simplex. Dirichlet distributions are a multivariate generalization of the beta distribution and are often used as a prior distribution. The three simplexes shown in Figure 12 are examples for the Dirichlet functions probability density functions used to classify concepts over a probability vector. We use expert knowledge similar to the simple concepts introduced in the beginning of this chapter to learn the parameters that define these distributions. Each of them corresponds to one of the three concepts described in the above list. Distribution *(a)* shows the concept sure about one characterized by a peak near c_1. Since the probability vector is sorted in descending order we know that the highest probability will always correspond to the corner c_1. Similar to the first concept, distribution *(b)* is characterized by a peak between the highest and second highest concept. The third simplex *(c)* shows the concept of being unsure about any of the three possible probabilities.



(a) Sure about 1 class (b) Either of 2 classes (c) Unsure

Figure 12: Dirichlet distribution simplexes for classifier concepts

The sample in the above figure is represented by a red dot. We can see how the sample is located close to the corner resembling the concept of "being sure about 1 class". To determine the values v_c for the probability vector parameter we use the probability density function of the 3 concept distributions as a classifier. We get the following values:

Concept	Raw PDF value	Normalized PDF (v_c)
Sure about 1 class	6.942	0.875
Sure about 2 classes	0.793	0.10
Unsure	0.195	0.025

By selecting the highest value, v_1, the sample is being classified as sure about one class.

4 Usability: A Pilot Study

To evaluate the functionality and the influence of the explanations generated by the explanation generation framework introduced in Chapter 3, we conducted a pilot study with experts in the field of machine learning and robotics. In the user study, the experts interacted with a simulated vacuum cleaning robot acting in a grid world. Using our explanation generation framework, the users were able to ask the robot for explanations for its actions. We analyzed if the explanations helped the participants to understand a robot's decision making better than a control group without the explanation feature. Furthermore we collected metrics on how the subject's interaction with the user interface was influenced by the availability of an option to request explanations.

4.1 Experiment Structure

The experiment consisted of 2 interactions for each of the two groups of participants. These two groups were a test group that had the possibility to request explanations, and a control group which did not. Figure 13 shows the structure of the experiment and which data is gathered during each step. Before the experiment started, the concept classifiers were learned from an expert as presented in Section 3.3.

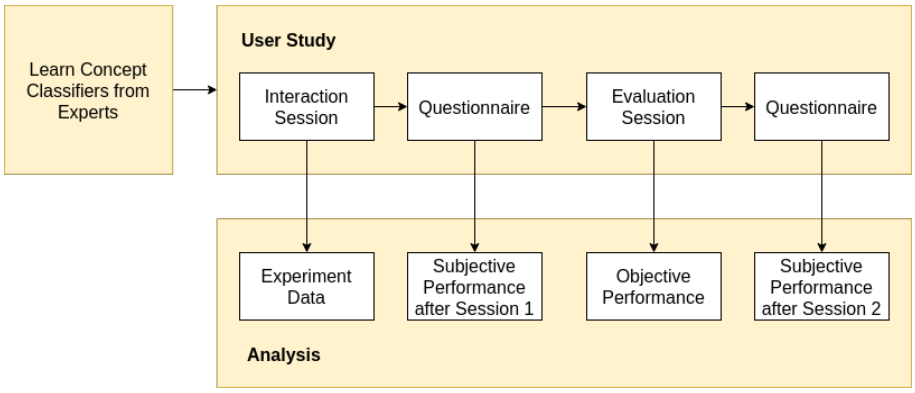


Figure 13: Structure of the experiment and collected data

In the first session, the interaction session, the participants were introduced to the experiment and provided with instructions printed on paper. During the interaction session, the subjects did not know that there would be a second session in the experiment. After the first session, the subjects filled a post-session questionnaire.

After finishing the questionnaire, the subjects were introduced to the second session, the evaluation session. This part of the experiment was designed similar to the interaction session. However, instead of observing the robot, the task was to reproduce the robot policy from the first session from the robot's perspective. Then the subjects filled a final questionnaire.

Goals The goal of this study was to compare the subjective performance of the test and control group with their relative performance. We wanted to know how well

the participants of each group think they performed in reproducing the policy. Furthermore we compared the subjective performance to a performance score calculated after the experiment. We investigated how well the groups perform relative to each other as well as how well one subject performs compared to its group. Based on the results of this comparison we investigated the influence of the explanations on the subjective and relative performance.

Participants In this pilot study the subjects were 6 robotics experts evenly divided in a test and control group. The subjects were on average 26 years old and within a range from 24 to 36 years. All of them were male. Experts in the field are familiar with the tools applied when developing the explanation generation framework. This introduces possible biases into the user study. Having experience in robotics puts the experts in a different initial situation than layman users. While a non-expert might see the robot as a black box, an expert has more experience that helps them to predict and analyze the robot behavior. Thus experts start with a different mental model of the robot. However using experts as subjects for a pilot study allows us to gather valuable information about the explanation generation method that non-experts may not be aware of. In further research we will use this feedback to optimize our method and design a larger user study target towards non-experts. The subjects were divided into two groups.

4.2 User Interface

For the experiment we designed a User Interface (UI) that allowed the subjects to interact with a vacuum cleaning robot in a grid world. For the different steps of the experiment we developed variations of the UI. Figure 14 shows the interface of the interaction step for the group that had the possibility to ask for an explanation. The UI was implemented in Python using Pygame, a python library for game development.

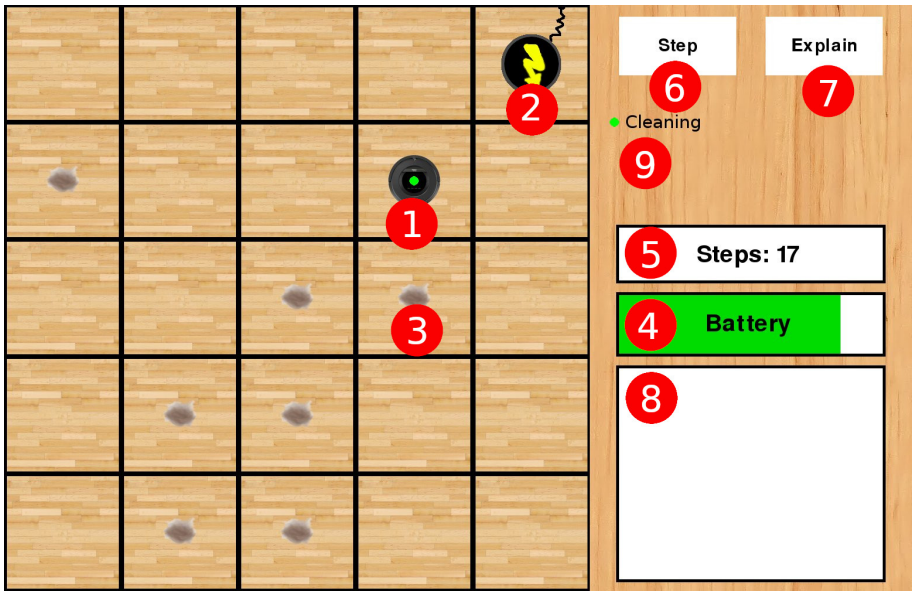


Figure 14: Experiment interface with enabled explanations

The grid world as shown in Figure 14 consists of 5×5 tiles T and represents the room in which the robot (1) is located. On the right side of the interface the menu for interacting with the interface is located. Each tile $t_{x,y}$ of the grid world can contain a dirt particle (3) d that indicates that it is dirty. A special case is the charging station (2) in state $t_{0,4}$. This tile can not get dirty and charges the robot. The resulting dirtiness D of the room is calculated as

$$D = \frac{1}{T} \sum_{d=1}^{T} 1 \tag{21}$$

The Robot We use a simulated vacuum cleaning robot. It has a policy with the dirtiness of the room D and the battery level B as parameters p_1 and p_2. The three possible actions are a_0, a_1 and a_2. The battery level B is indicated in the UI by the green bar (4). The policy of the robot is shown in Figure 15. The current state of the robot is indicated by an LED (9) on top of it and with a description in the user menu. For this experiment we used a policy with 3 possible actions $a_0 = \textit{clean}$, $a_1 = \textit{charge}$, and $a_2 = \textit{stop}$. If the robot is in the *clean* state it will move to the closest dirt particle in the grid world. If it is in *charge* state it will move to the charging station. The charging process is instantaneous when the robot arrives at the charging pad to save time. In the *stop* state the robot will stop in place and not discharge its battery.

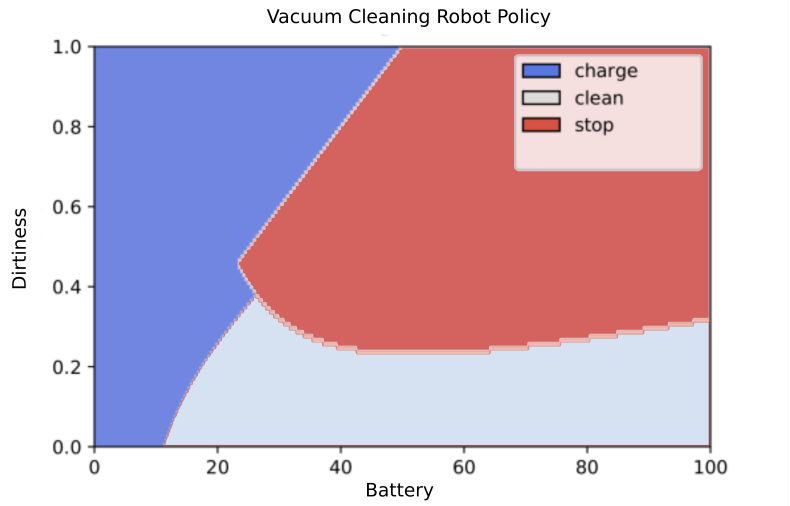


Figure 15: Erroneous vacuum cleaning robot policy

The Policy Figure 15 shows the policy. In the experiment we do not use an optimal cleaning policy. We inverted the clean and stopping region so that the robot will stop for high dirtiness and clean in low dirtiness. By phrasing the task as a debugging session, we remove the bias that participants have towards cleaning robots from previous experience or from common sense.

The Environment The environment is based on discrete steps If the user presses the "Step" button (6), the interface calculates the next state s' based on the current state s and the policy π. The environment then updates the battery level B and the dirtiness D of the room based on the action the robot took. Depending on the action that the robot took, its position in the environment is updated. Every step it can move 1 tile up, down, left or right in the grid world. A step counter (5) shows how many steps the current interaction episode lasted. If the user presses the "Explain" button (7), the robot generates an explanation and presents it in the output field (8).

The UI version without the ability to explain actions is identical to the one presented in Figure 14 except the explanation button was removed. For learning the concept classifiers and the evaluation session the structure of the UI was kept but the user menu was restructured.

4.3 Learning Concept Classifiers

As introduced in Section 3.3, the first step is to learn concept classifiers from experts. To keep this labeling process similar to the experiment scenario, we implemented the process using a variation of the user interface introduced in Section 4.2. The concept classifier interface is shown in Figure 16.

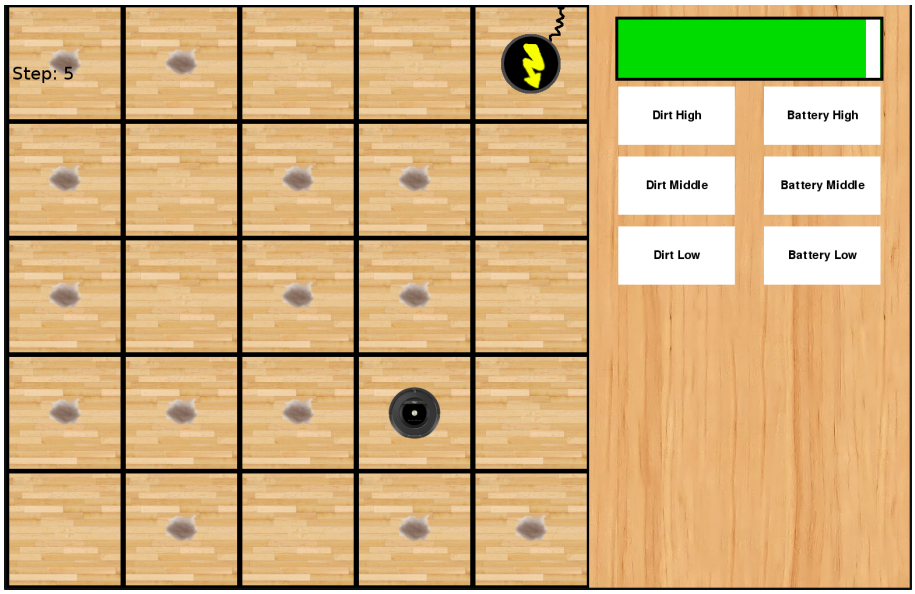
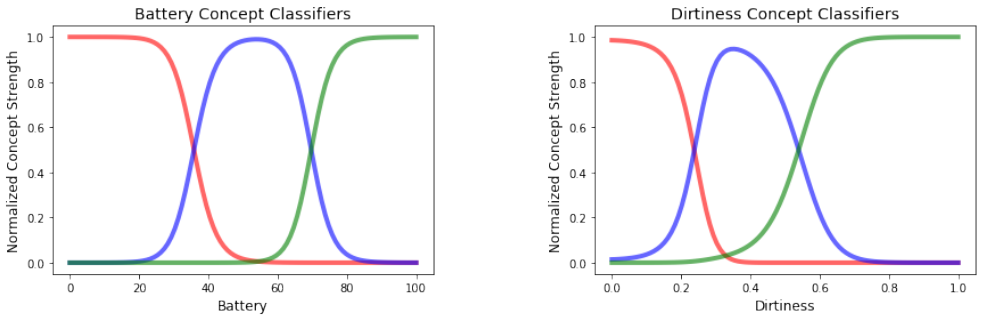


Figure 16: Concept classifier labeling user interface

For the labeling process, we provide a button for every concept C_p in every parameter P. The buttons are displayed in the user menu on the right side of the UI. A random sample s of dirtiness D and robot position $R_{x,y}$ in the grid world as well as battery charge B is generated. The sample is labeled by pressing one button in the interaction area for each parameter. In the particular scenario in Figure 16 the battery level and dirtiness are subjectively high and thus the button "Dirt High" and "Battery High" label this state sample accordingly. The step is considered fully labeled when all parameters got one label and the next state is generated for labeling. To generate the concept classifiers for this experiment we used 50 samples.

For this experiment we chose three concepts for each parameter because it allowed us to represent the policy of the robot while not making the explanations too complex.

The resulting normalized concept classifiers are shown in Figure 17. The classifiers were saved for later use in the interaction part of the experiment.



(a) Concept classifier for the battery parameter

(b) Concept classifier for the dirtiness parameter

Figure 17: Normalized concept classifiers.

4.4 Session 1: Interactions with the Experiment Interface

In the first session, the interaction session, the subjects interacted with the simulated vacuum cleaning robot through the UI introduced in Section 4.2. The subject's objective was to learn the rules of the robot's decision making or the policy. Additionally we logged metrics in order to analyze performance and behavior of the subjects. The metrics are presented in detail in Section 4.4.2. During the first session, the subjects did not know that there would be a second session. We did not provide this information because the subjects' behavior in this session might have been biased if they were aware of the next session in which they had to reproduce the robot policy.

4.4.1 Instructions

The subjects received instructions printed on paper. The scenario described in the instructions was that the subject had to debug a vacuum cleaning robot that behaved unexpected. The instructions shown in Figure 18, were phrased to be a descriptive story in order to keep the subjects involved in the situation.

After explaining the user interface from Section 14, the subjects were reminded that they could end the experiment at any time. Furthermore the subjects had the possibility to reset the UI to a random state of robot position, battery level and dirtiness. This option was part of the experiment because the erroneous policy could cause the robot to get stuck. For debugging it is important to have the possibility to restart the scenario in order to be able to investigate the results of different starting situations.

After the subjects were familiar with the experiment setup, the experiment could be started by pressing the "Step" button for the first time.

4.4.2 Metrics

The metrics recorded during this session were the following:

- Session duration
- Number of steps taken
- Number of explanations requested

The session duration was tracked because the user patience can vary and we assume the understanding of the robot policy increases with the time spent on interacting with the robot. The number of steps was tracked for the same reason as the session duration. Furthermore, the number of steps taken in relation to the session duration allows us to understand how long a user was reflecting during each step. The above metrics are logged to investigate the different approaches that subjects take on the debugging. We record the battery level and dirtiness of the room for each of the steps and at which step explanations were requested. These metrics will be used to test the following hypotheses:

You are the robot specialist of a company that is producing vacuum cleaning robots. The latest robot model had to be recalled because customers complained about the robot not cleaning properly and "doing weird things". It is your job to find out what is wrong with the robot's behavior.
You know that the robot is equipped with sensor that can detect how much dust is in the room. The robot also knows how much battery is left. Based on this information, the robot decides whether or not it continues to clean, stops and waits in place or if it will go back to the charging station. An LED on the top of the robot indicates which of these 3 actions the robot just took.
After you tested all sensors and the robot's ability to navigate around in the room you know that these functions work correctly and need no further investigation. The only thing left that could cause an error is the robot's logic for when he will clean, go to charge or wait in place. You decide that your next test will be to debug the robot's decision making when to execute these 3 actions. In order to debug the robot's logic you can use the following tools:

- *The [Step] button to make the robot execute one more step. For example if the step button is pressed and the robot is in cleaning mode then it will move to clean the closest dust pile. If the robot is stopping, it is switching to energy saving mode and waiting in place. In the charging mode the robot will move to the charging station and recharge its battery when it reaches the station.*
- *The [Explanation] button allows you to ask the robot for an explanation of why the last action was performed. For example if the robot unexpectedly goes to charge its battery you could ask for an explanation why the robot went to charge. The explanation will show up in the white explanation box in the user interface.*
- *If the robot crashes or is stuck you can reset the environment to a random state by pressing 'r' on the keyboard.*

Figure 18: Instruction and background for Session 1

1. Longer session duration and a higher number of steps increase the performance when reconstructing the policy in the second session
2. The number of explanations that a user requested increases the performance when reconstructing the policy
3. The group that has access to the explanation generation framework can estimate the understanding of the policy better than the control group
4. The group that has access to the explanation generation framework performs better than the control group when reconstructing the policy

4.4.3 Questionnaire

After the interaction part ended, the subjects filled a questionnaire. Next to basic demographic questions, the questionnaire contained the following three Likert-scale statements:

1. I fully understand the robot's behavior
2. The explanations were useful
3. I am confident to explain the robot's behavior to someone else

Furthermore the questionnaire asked the reason the participants chose to end the session with the following options:

4. I think I learned enough about the robot's behavior
5. The interaction was boring
6. I was confused by the robot's behavior
7. I was not able to get more information about the robot's behavior with the given tools
8. Other

Lastly, the subjects were asked to describe their strategy when asking for explanations.

The control group received the same questionnaire without question 2. All subjects were asked to give explanations for their marks on the Likert scale.

4.5 Session 2: Reconstructing the Robot's Behavior

The subject's objective in this session was to reproduce the erroneous robot's behavior. The policy π_1 used in the first session was a SVM. We collect samples S where the user provides a mapping from the current state sample $s \in S$ to action a. The user mapping U is given by Equation 22, the control mapping T is calculated using the policy π_1 to determine the action of the label according to the correct policy.

$$U = \begin{pmatrix} s_1 & a_1 s_2 & a_2 ... & ... s_S & a_S \end{pmatrix} \tag{22}$$

$$T = \begin{pmatrix} s_1 & \pi(s_1) s_2 & \pi(s_2) ... & ... s_S & \pi(s_S) \end{pmatrix} \tag{23}$$

Then we used the Jaccard index (Equation 24) and a measure of correctly labeled samples to the total number of samples 25 to calculate the performance measures. The number of correctly labeled samples $\overline{s}$ is calculated. Furthermore we calculate the Jaccard index JS. These metrics help us to analyze the performance of the subject when reproducing the policy.

To label the state space configurations, the subjects interact with a variation of the UI that was used in the previous session. Figure 19 shows this configuration.

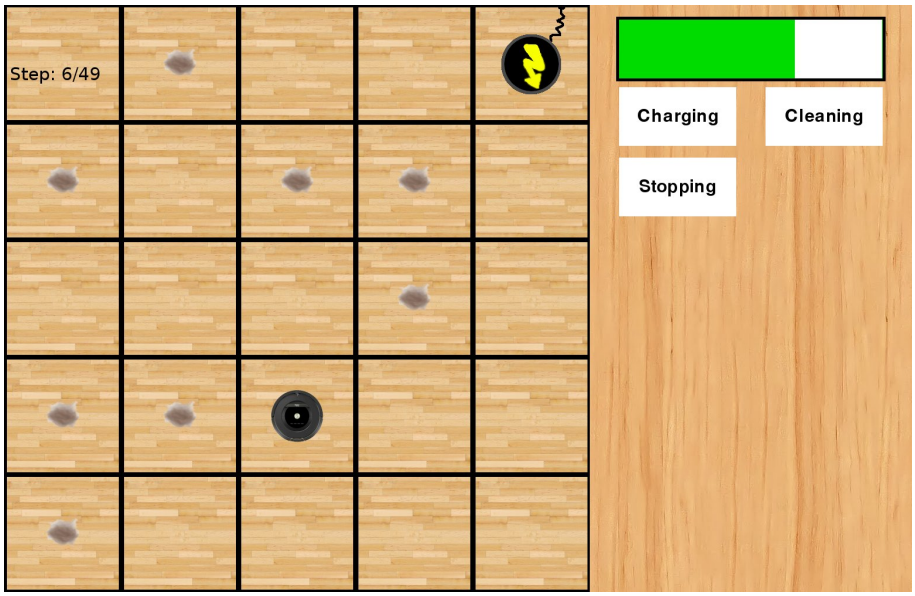


Figure 19: Reconstructed Policy from 49 evenly spread samples

Overall the subjects labeled 49 samples that evenly cover the state space. We chose 49 samples because we could provide sufficient coverage of the state space with 7×7 samples while keeping the user effort relatively low. The order in which the samples were to be labeled was randomized. Furthermore the instructions stated that all steps are randomly generated to prevent bias towards the reconstruction method.

4.5.1 Instructions

The subjects were receiving a new set of instructions on paper shown in Figure 20. The subjects are instructed to reproduce the erroneous policy of the robot. The instructions emphasized that the subjects had to reproduce the erroneous policy from the previous session. This was taking account for possible biases regarding the functionality of vacuum cleaning robots that could lead subjects to reproduce a new, working policy.

Now we will test your understanding of the robot's decision making. The goal of this task is to reproduce the faulty behavior of the robot based on what you learned about the robots behavior in the previous step. You will see the same user interface as before, but now every step will be a RANDOM generated state of the room and robot. You will be provided with the battery state of the robot and you will also see how much dirt is in the room. Your task is to describe what the faulty robot would do next in the current situation and what you previously learned about the robot's behavior. Please report what the faulty robot WOULD do not what you think the robot SHOULD do. If you think that the robot will go to clean in the next step press the [Cleaning] Button and so on.

Figure 20: Instruction and background for Session 2

The user menu has been replaced by buttons to provide the action labels for the current state space configuration. A configuration was presented in the same way as before by the battery indicator and the dirtiness of the room. If the user labeled the current configuration by pressing one of the buttons, the next configuration was presented. The session ended when the subject labeled all 49 samples.

4.5.2 Metrics

At the end of this session we calculate the relative performance that each participant has achieved compared to his group. We compare the similarity between the original and the reproduced policy π_1 and π_2. To calculate similarity measures we use the Jaccard index between two sample sets generated using the policy SVM and the labeled samples.

The Jaccard Index is a measure for the similiarity between to finite sets. The Jaccard Similarity Index JS for two finite sets A and B is calculated as

$$JS(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{24}$$

Additionally we calculate how many of the labeled samples from session two were conforming with the original policy. A sample that has the same state to action mapping as the original policy π_1 is considered as correctly labeled. The ratio r_p of correctly labeled samples $\overline{s}$ to all samples $S = 49$ is calculated as

$$r_p = \frac{\overline{s}}{S} \tag{25}$$

We expect that the group that has access to the explanation feature will perform better regarding the calculated metrics than the control group.

4.5.3 Questionnaire

After finishing the second session the users filled another questionnaire containing Likert-scale statements. The questionnaires had one statement for both groups: "I

think I was able to fully reproduce the robot's behavior". Furthermore, the subjects were asked to give explanations for their marks on the Likert scale.

Additionally the group that had access to the explanation feature was asked:

1. The robot's explanations from the first session of the experiment helped me in this session

2. I feel generally confident about collaborating with a robot that can explain its decision making to me

The control group was asked "Do you think it would be beneficial if the robot would be able to explain its decision making to you? (e.g.: "I did clean because it was very dirty.")"

5 Results and Discussion

In this chapter, we analyze the data collected during the user study. The six subjects were divided evenly into a test group that could query the explanation generation framework and a control group that was not able to request explanations. We first compare the subjective performance for each group. The subjective performance describes how well the subjects think they performed in the task to reproduce the robot's policy. Then we discuss and compare the objective performance for each group based on the Jaccard index and the number of correctly labeled states. The objective performance describes how well the subjects performed compared to their own group and the other group. These results allow us to analyze how the possibility to request explanations improved the performance, as well as the subjects predictions towards their performance. We also analyze how the differences in session duration and the number of steps and explanations influences the performance metrics. After that we analyze the feedback given in the questionnaires.

Lastly, we present how the results of the pilot study will be used to improve the experiment design and the explanation generation framework for a future user study.

5.1 Subjective Performance

First we analyze and discuss the subjective performance for each group. The subjective performance describes how the subjects predicted and rated their own performance. The feedback for questions and the results of the Likert scale statements can be found in Appendix A and B. In Table 1, 2 we compare the average results of both groups for each questionnaire.

Statement	Test Group	Control Group
I fully understand the robot's behavior:	2.33	2.00
The explanations were useful:	2.33	-
I am confident to explain the robot's behavior to someone else:	4.33	3

Table 1: Questionnaire 1

Statement	Test Group	Control Group
I think I was able to fully reproduce the robot's behavior:	3.66	2.33
The robot's explanations form the first session helped me in this session:	1.33	-
I feel generally confident about collaborating with a robot that can explain its decision making to me:	4.00	-
I think it would be beneficial if the robot would be able to explain its decision making to me:	-	6.33

Table 2: Questionnaire 2

As can be seen from Tables 1 and 2, the subjective performance of the test group was slightly better than for the control group. However both groups reported that they do not fully understand the policy and the test group rated the usefulness of the explanations low with an average value of 2.33 on the Likert-scale. For the test group this was surprising because in two cases subjects of this group voiced a correct description of the erroneous policy during the experiment: "It does stop when it's dirty and clean when it's not dirty, right?". When analyzing the feedback for question 2 in the first questionnaire, in two cases the subjects described that "only the charging explanations made sense" and "The explanations did not really help. Why would the robot not want to clean even the nearest grid cells if the battery is full and the room is very dirty?". This described the purposely faulty policy. The part of the policy that led to the robot taking the charging action was intuitively identified and the robot did not clean when the dirtiness was high. We therefore assume that the experts did identify the error in the policy, but did not trust their understanding. Another reason for this disparity might be that the subjects projected the erroneous policy to the correct explanations. Since the explanations correctly explain the wrong policy the subjects assumed the explanations to be wrong or misleading as well. Furthermore, the experts might have higher expectations towards the explanations. As mentioned in [19], answering the "Why?" intelligibility question provides adequate information but if the users have higher expectations towards the complexity of the explanation, they might be more unsatisfied with the explanations.

As expected, in the control group the self perceived understanding of the policy was rated low with a an average value of 2.0 on the Likert-scale. Furthermore the subjects of the control group did not feel as confident as the test group to explain the policy to someone else. Two subjects explained that the robot's behavior was not intuitive and one subject assumed that the robot's behavior was not related to the battery and dirtiness parameters. Another subject stated that the robot would be stuck forever once it enters the stopping mode. This is a correct observation, but similar to the other subjects that discovered the error in the policy, the connection that this behavior was part of the faulty policy was not established. We believe that the instructions need to be more specific when introducing the robot, its functionality and possible defects to the subjects.

Hypothesis 4 could not be confirmed. Against our expectations, the average objective performance as shown in Table 3 was higher for the control group. However these results are not allowing further analysis due to the low number of subjects in each group. We believe that the better performance of the control group can be explained by their higher session duration. The influence of of the session duration on the performance is discussed in Section 5.2.

Confidence to Explain the robot's behavior From Table 3 we can see that the confidence to explain the robot policy was higher for the test group. In the control group only one participant stated to be confident to explain the policy because the robot was following simple rules. These results were expected, however the small number of subject does not allow for further analysis of hypothesis 3. Noteworthy is that two subjects rated their confidence to explain the robot's behavior with a 6 on the Likert-scale but the feedback to this question generally indicated high doubt about the understanding of the policy. We assume that the voiced lack of confidence is a result of confusion caused by the explanations of the erroneous policy.

5.2 Comparing Subjective and Objective Performance

The objective performance refers to the data collected from the second session to measure how well the subjects reconstructed the erroneous policy. This and general data about the experiment is displayed in Table 3.

Metric	Test Group	Control Group
Average Session Duration in seconds	368	492
Average Session Duration in steps:	188	366
Average Number of explanations requested:	24	-
Average Jaccard Index:	0.49	0.57
Average correctly labeled samples:	23	27

Table 3: Average measured metrics

The metrics to measure the reconstruction performance are the Jaccard Index JS (Equation 24) and ratio r_p (Equation 25). We expected better performance for the test group because of the explanations, but according to the average values for JS and r_p, the control group performed better. However this is probably due to the fact that one subject reproduced an intuitively correct vacuum cleaning policy instead of the erroneous one which influenced the averages for JS and r_p due to the small sample size.

Furthermore hypothesis 1, that longer interaction time will lead to improved performance for both groups, could be the reason for the better performance of the control group. The average number of steps was about twice as high as for the test group and average interaction session for the control group was 33% longer. The longer interaction time allowed the user to become more familiar with the robot's behavior. Combined with the fact that the subjects generally failed to connect the

faulty behavior to the faulty policy, we believe that not having explanations but more time to learn about the behavior is better than having explanations. However, we attribute this confusion mostly to the way the instructions were phrased and the user interface was designed. Another reason for these unexpected performance differences might be that the experiment is subject to random effects. The amount of dirt and the position in the grid world is random.

Furthermore when the subjects reset the UI a new random state was initialized. For example the robot usually never reached 0% battery because it would go to charge before that. Also we limited this random re-initialization to an area of the state space in order to create more scenarios that will lead to discovering the error in the policy. Although we tried to account for this by limiting the possibilities to get uninformative states upon re-initializataion, we assume that for subjects that spent fewer steps on the interaction session experienced less of the state space and were more prone to receive uninformative random states. For example if the subject resets the environment two times, the chances are higher that the robot gets stuck immediately and it is not possible to tell when this behavior starts to occur. This might also explain why the subjects in the control group performed better that the test group. Spending more time in the interaction step not only increases the area of the state space covered, but also reduces the effects of randomness in the experiment.

Next we analyzed the behavior of the subjects regarding the time and frequency of explanation generation. The three subjects in the test group used two different approaches for requesting explanations. One approach involved asking for an explanation when the robot's behavior was unexpected or unfamiliar. The other strategy was to request an explanation after every step. Comparing these approaches does not allow further analysis because of the small number of subjects.

Hypotheses 2 and 4 could therefore not be confirmed. However we think that by removing the biases, simplifying the UI and clarifying the instructions an experiment with more subjects will confirm hypothesis 4. Hypothesis 2 will require a study with more subjects to make statements about the performance of different strategies and analyze how the number of explanations influences the performance.

5.3 Analyzing Questionnaires

In this section we will discuss the feedback that is not related to the subjective and objective performance of the subjects.

5.3.1 Feedback

One subject in the control group mentioned that the feedback through the status indicator LED on top of the robot was helpful when observing the robot behavior. We assume that the control group used the LED as a transparency mechanism to observe the behavior of the robot. The LED might have contributed to the performance of the control group because for that group it was the indicator for the action of the robot.

Analyzing the feedback on question 2, the usefulness of the explanations, we could confirm our theory that participants did not fully understand the context of the explanations. One subject thought that the concept selected for the explanations indicated that the robot made its decision only on the parameter described by this concept. This shows that the instructions were not precise enough in defining how the explanations are generated. Another explanation is that experts are biased by their expectations towards the complexity and structure of the explanations because of their experience with robot decision making. A subject of research will be to rework the instructions to be more understandable and to define the problem and tools more clearly.

When analyzing the reason for ending the experiment, the interaction was deemed boring by four out of six subjects. One explanation for this feedback is again the bias of the experts and their higher expectations towards the explanations. Furthermore in their paper on the importance of explanations [4], Bunt et al. found that the users interest in explanations depends on the control that the user has over the agent [17]. In our scenario the user is observing the robot without any control over it and thus might be subject to the same effect as discovered by Bunt et al.

5.4 Discussion

The goal for further research is to conduct a larger user study with a group of non-experts. In this section we will describe how the findings from the pilot study can be used to improve the explanation generation framework and the structure of the experiment.

Using the same settings for the experiment makes it possible to repeat the experiment with the same conditions. The explanation generation framework is general regarding the policy and no constraints are given for the number of parameters. For example, it is possible to use the explanation generation framework with state spaces that have more than 2 parameters or robots with other continuous polices and discrete actions. Similarly, the number of concepts is flexible.

5.4.1 Improving the Explanation Generation

Based on the analysis of the user feedback we, identified 3 areas for improving the explanation generation framework:

1. Deciding when to stop increasing the sampling radius and generate an explanation

2. Calculating the concept ratios in corner cases of the policy

The Sampling Process The sampling process is a crucial component of the explanation generation framework. During tests of the framework we made the following assumptions:

1. The parameters are finite

2. When covering a certain area of the state space, we can be sure to find the policy and generate an explanation with weighted parameters

However the example in Section 3.4.3 has shown that there are cases for which the explanation generation does not provide reliable explanations. For the follow-up study we will improve the explanation generation framework by changing the stopping conditions. Instead of sampling until we either reach a set threshold or the policy boundary we sample until we find the policy boundary and then increase the sample are to make sure that a certain amount of samples is in a region of the policy with a different action. For example, we could set the stopping condition so that we stop increasing σ when we can be 99.7% sure that 10% of all samples are over the policy border and have a different action. By doing this we make sure that the parameters get always weighted correctly. For some state space and policy combinations this approach might require to sample large parts of the state space, but we believe that weighting the parameters is more important for intelligible explanations than explaining as local as possible. For most regular cases this improvement would prevent wrong explanations for parameter values far away from the policy boundary. However, further testing with this method for these corner cases is required. Furthermore, this approach is relying on the assumption that the parameters are finite. If the parameters would not be finite, it is possible be that we never find the border or we have to sample too long to find the policy boundary.

Concept Ratios for Corner Cases Further corner cases that could be improved are areas in the state space in which the policy is more complex. Complex in this case means, for example, non-linear policies or regions where multiple policy boundaries cut each other. In the current implementation the weighting of concept ratios for more than two concepts uses the average Jensen-Shannon distance between all histograms (See Equation 19). But by averaging we lose information about the course of the policy border. During this experiment the influence of this effect was minor because the policy was not complex and except one crossing of policy boundaries, the policy was mostly linear. However, more complex policies will require a new scaling mechanisms.

5.4.2 Follow-up Study

Next to the above improvements to the method, we will also take the findings of the experiment feedback and our analysis to optimize the experiment into account. We identified the following problems:

1. The instructions were not precise or not complete enough and confused the subjects regarding the context of the explanations
2. The number of subjects in the experiment was small
3. The experiment scenario was subject to random factors
4. The vacuum cleaner scenario introduced a bias

Problem 1 First a problem that has been evident throughout the analysis was the confusions towards the context of the explanations. Subjects seem to project the erroneous policy to the explanations and as a result doubt the correctness and usefulness of the explanations. The instructions therefore require more precise descriptions of what the explanations are explaining and how they can be used to debug the policy. For the follow-up study we will rephrase the instructions with more detail about what the purpose of the explanations is and use more examples as demonstrations. Also we will stress that the explanations are always correctly describing the robot behavior and that unexpected explanations might indicate a faulty policy. For example a section in the instructions about how the explanation framework has previously been used to debug a similar error in the policy could be added.

Problem 2 This study was a pilot study involving 6 subjects. The small number of subjects does not allow a in depth analysis of the resulting metrics regarding the performance. However the experiment structure will be improved and future large scale study involving non-experts will solve this problem.

Problem 3 The random re-initialization and randomized dirt creation are introducing randomness into the experiment. This problem will be mitigated by increasing the number of subjects. However we could remove this factor by defining a sequence of states for the re-initialization that are in the same order for every subject.

Problem 4 When designing the experiment we introduced an error in the policy of the robot to make up for bias introduced by common sense and previous experience. We assumed that humans already have a strong mental model of what a vacuum cleaner is supposed to do. Therefore, asking to reproduce a correct cleaning policy could be solved by applying previous knowledge about cleaning processes to the labeling process. One subject reproduced an intuitively correct cleaning policy based on their interpretation of how a cleaning robot should work, not on based on the purposely faulty policy. This can be attributed to misunderstanding regarding the instructions, but another way of mitigating this problem is to use a different scenario for future studies. The scenario should be one that has either no reference to a real world scenario or a scenario that is not common. For example the scenario could be rephrased to debug a mars rover that is collecting material samples.

6 Conclusion

The aim of this thesis was to investigate the current state of the research regarding transparency in HRI and based on previous research, investigate how explanations can help to enhance interactions between humans and robots. Based on previous research on policy explanation and explanation generation, we developed an explanation generation framework for automated local policy explanation. We designed and conducted a pilot study with the goal of improving the framework and experiment for a future large scale user study.

The explanation generation framework introduced in Chapter 3 uses concepts, expert learned natural language labels attached to a region in the state space. These concepts are represented by concept classifiers which are learned from an expert that is familiar with the robot and its policy. By sampling in the state space, the framework calculates a measure for the strength of the concepts in the vicinity of the robot state by using the concept classifiers. To determine which parameters influence the policy in the sample region, we developed a method of weighting these concept strengths depending on the related parameters. Using the information about the relevant concepts and their natural language representation, the framework selects the strongest concept and generates the explanation that answer the "Why?" intelligbility question introduced by Lim et Al. [20, 19].

In the experiment conducted in Chapter 4, we compared the perceived performance and compared it to the objective performance for a small group of subjects consisting of experts in the field of robotics. We found out that longer interaction with the robot leads to better performance when reproducing a robot's policy independent of the explanation capability of the robot. Our hypothesis that explanations improve the performance as well could not be confirmed. However, this might have been the result of the following limitations of the experiment design: First the feedback and metrics gathered during the experiment indicated that the subjects did not always understand the instructions or trusted their own mental models more than the explanations. Also the subjects seemed to transfer the faultiness of the policy to the explanations and assumed that they were faulty too and thus deemed them useless. To improve on the experiment we will rework the experiment instructions with clearer instructions regarding the context of the explanations. To remove biases resulting from the experiment scenario we will change the scenario to one that possible subjects are not familiar with. Second we will take further improvements on the explanation generation framework. The user study showed that the sampling process in the explanation generation framework performed worse in cases when the sampling process did not detect a policy.

With these improvements we will conduct a follow-up study with a larger group of subjects comprised of non-experts. This will allow us to gather statistically significant data and get reliable measures for how the explanations enhance the subjects ability to debug the robot and reproduce its policy.

Transparency in HRI is still a new field and especially in the area of automated explanation generation little research has been done. However we think that trans-

parency and explicability will help to shed light into the black box that robot decision making has been so far, especially for the layman user. Therefore, making robots and intelligent agents transparent will be a necessary step towards widespread adoption of robots in everyday life.

References

[1] Saleema Amershi, Maya Cakmak, William Bradley Knox, and Todd Kulesza. Power to the people: The role of humans in interactive machine learning. *AI Magazine*, 35(4):105–120, 2014.

[2] Victoria Bellotti and Keith Edwards. Intelligibility and accountability: human considerations in context-aware systems. *Human–Computer Interaction*, 16(2-4):193–212, 2001.

[3] Joanna J Bryson. The meaning of the epsrc principles of robotics. *Connection Science*, 29(2):130–136, 4 2017.

[4] Andrea Bunt, Matthew Lount, and Catherine Lauzon. Are explanations always important?: a study of deployed, low-cost intelligent interactive systems. In *Proceedings of the 2012 ACM international conference on Intelligent User Interfaces*, pages 169–178. ACM, 2012.

[5] Maya Cakmak, Crystal Chao, and Andrea L Thomaz. Designing interactions for robot active learners. *IEEE Transactions on Autonomous Mental Development*, 2(2):108–118, 2010.

[6] Maya Cakmak and Andrea L Thomaz. Designing robot learners that ask good questions. *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*, pages 17–24, 2012.

[7] Maya Cakmak and Andrea L Thomaz. Eliciting good teaching from humans for machine learners. *Artificial Intelligence*, 217:198–215, 2014.

[8] Mark W Craven and Jude W Shavlik. Using neural networks for data mining. *Future generation computer systems*, 13(2-3):211–229, 1997.

[9] Joachim de Greeff and Tony Belpaeme. Why robots should be social: Enhancing machine learning through social human-robot interaction. *PLoS one*, 10(9):e0138061, 2015.

[10] F Elizalde, E Sucar, and P DeBuen. A prototype of an intelligent assistant for operator's training. In *International Colloquium for the Power Industry. México: CIGRE-D2*, 2005.

[11] Francisco Elizalde, L Enrique Sucar, Manuel Luque, J Diez, and Alberto Reyes. Policy explanation in factored markov decision processes. In *Proceedings of the 4th European Workshop on Probabilistic Graphical Models (PGM 2008)*, pages 97–104, 2008.

[12] Julia Fink. Anthropomorphism and human likeness in the design of robots and human-robot interaction. In *International Conference on Social Robotics*, pages 199–208. Springer, 2012.

[13] Peter A Hancock, Deborah R Billings, Kristin E Schaefer, Jessie YC Chen, Ewart J De Visser, and Raja Parasuraman. A meta-analysis of factors affecting trust in human-robot interaction. *Human Factors*, 53(5):517–527, 2011.

[14] Bradley Hayes and Julie A. Shah. Improving robot controller transparency through autonomous policy explanation. In *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, HRI '17, pages 303–312, New York, NY, USA, 2017. ACM.

[15] Todd Kulesza, Margaret Burnett, Weng-Keen Wong, and Simone Stumpf. Principles of explanatory debugging to personalize interactive machine learning. In *Proceedings of the 20th international conference on intelligent user interfaces*, pages 126–137. ACM, 2015.

[16] Todd Kulesza, Simone Stumpf, Margaret Burnett, and Irwin Kwan. Tell me more?: the effects of mental model soundness on personalizing an intelligent agent. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1–10. ACM, 2012.

[17] Todd Kulesza, Simone Stumpf, Margaret Burnett, Sherry Yang, Irwin Kwan, and Weng-Keen Wong. Too much, too little, or just right? ways explanations impact end users' mental models. In *Visual Languages and Human-Centric Computing (VL/HCC), 2013 IEEE Symposium on*, pages 3–10. IEEE, 2013.

[18] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.

[19] Brian Y Lim and Anind K Dey. Assessing demand for intelligibility in context-aware applications. In *Proceedings of the 11th international conference on Ubiquitous computing*, pages 195–204. ACM, 2009.

[20] Brian Y Lim, Anind K Dey, and Daniel Avrahami. Why and why not explanations improve the intelligibility of context-aware intelligent systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2119–2128. ACM, 2009.

[21] Jianhua Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information theory*, 37(1):145–151, 1991.

[22] Joseph B Lyons. Being transparent about transparency. In *AAAI Spring Symposium*, 2013.

[23] Edward J McCluskey Jr. Minimization of boolean functions. *Bell system technical Journal*, 35(6):1417–1444, 1956.

[24] Lindsay M Oberman, Joseph P McCleery, Vilayanur S Ramachandran, and Jaime A Pineda. Eeg evidence for mirror neuron activity during the observation of human and robot actions: Toward an analysis of the human qualities of interactive robots. *Neurocomputing*, 70(13-15):2194–2203, 2007.

[25] Mattia Racca and Ville Kyrki. Active robot learning for temporal task models. In *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, pages 123–131. ACM, 2018.

[26] Byron Reeves and Clifford Ivar Nass. *The media equation: How people treat computers, television, and new media like real people and places.* Cambridge university press, 1996.

[27] Kartik Talamadupula, Gordon Briggs, Tathagata Chakraborti, Matthias Scheutz, and Subbarao Kambhampati. Coordination in human-robot teams using mental modeling and plan recognition. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 2957–2962. IEEE, 2014.

[28] Andreas Theodorou, Robert H Wortham, and Joanna J Bryson. Why is my robot behaving like that? designing transparency for real time inspection of autonomous robots. In *AISB Workshop on Principles of Robotics*. University of Bath, 2016.

[29] Anna-Lisa Vollmer and Lars Schillingmann. On studying human teaching behavior with robots: a review. *Review of Philosophy and Psychology*, pages 1–41, 2017.

[30] Ning Wang, David V Pynadath, and Susan G Hill. The impact of pomdp-generated explanations on trust and performance in human-robot teams. In *Proceedings of the 2016 international conference on autonomous agents & multiagent systems*, pages 997–1005. International Foundation for Autonomous Agents and Multiagent Systems, 2016.

A Subject Feedback

Question 1: I fully understand the robot's behavior. Please explain:
The robot did not do what I thought it would do. (I.e. it only cleaned seldom)
The robot tends to stay at the charging station even when the room is completely dirty. Explanations don't really help at this point.
I would need to do more experiments and draw a decision making tree.
Question 2: The explanations were useful. Please explain:
Most of the time the explanations did not make sense ("I did not clean because the dirtiness was medium"). Only the explanations when it went back to charge were useful/understandable.
When the robot is stuck at the charging station even after 500 steps and the explanation says Thinking, it does not really help explain why would the robot not want to clean even the nearest grid cells given that the battery is fully charged and the room is very dirty.
The whole state space was not mapped into the action in the explanation. Only one state was printed and based on that it was decided what robot is going to do.
Question 3: What was your strategy when to ask for an explanation?
Save as much energy as possible, i.e. not move. Only clean if it is on the same field or next to it.
I would like for an explanation when the robot preferred to go to charging station and stay there for over 20 steps and moreso when the room was quite dirty.
Step-explanation-step-explanation-...
Question 4: I am confident to explain the robot's behavior to someone else. Please Explain:
I observed the robot long enough
All I understood about the programmed behavior was that the robot preferred to stay at charging station rather than cleaning the room which I am pretty sure was not the intended purpose. I understand that the robot does not take actions based off on positions but simply based on battery charge and amount of dirt, but I failed to understand why it would like to get stuck at the charging station despite multiple restarts.
It was random

Table A1: Questionnaire 1, Test Group

Question 1: I fully understand the robot's behavior. Please explain:
Seems unintuitive; the behaviour didn't seem to be related to the position of dust and battery state that much
Because the LEDs give a good feedback of the current state
I could not work out what made the robot go in to waiting mode, or why the robot stayed in waiting mode, apparently forever, once it started
Question 2: I am confident to explain the robot behavior to someone else. Please explain:
Because it's a simple robot.
Like many things in life, it kind of works, but has unexplained behaviour. That is quite normal.

Table A2: Questionnaire 1, Control Group

Question 1: The robot's explanations from the first session of the experiment helped me session:
The explanations were not detailed enough with respect to the overall goal of the robot. It should include a reason for its current action that explains for example "I did not clean because I only clean when the dirtyness is high or higher"...
The explanation was not complete in the first session.
Question 2: I feel generally confident about collaborating with a robot that can explain its decision making to me:
Some of the explanations were good (with regard to charging) but others were a bit confusing (with regard to cleaning)
As long as I know what the robot has planned I can plan my work around its behavior to ensure a smooth collaboration with minimal conflicts.
I saw programs controlling the robot. They should not be trusted.
Other Feedback:
Instead of completely random states, consider some heuristic parameter setup so that the environment settings are more realistic.

Table A3: Questionnaire 2, Test Group

Question 1: I think it would be beneficial if the robot would be able to explain it's decision making to me (e.g.: "I did clean because it was very dirty."):
The behaviour seemed random
Because that might provide additional information on why the faulty behavior occurred.
Nobody trusts robots because their "mental" processes are inscrutable

Table A4: Questionnaire 2, Control Group

B Experiment Data

	Scores	Average
I fully understand the robot's behavior:	3,3,1	2.33
The explanations were useful:	4,2,1	2.33
I am confident to explain the robot's behavior to someone else:	6,6,1	4.33

Table B1: Questionnaire 1, Test Group

	Scores	Average
I fully understand the robot's behavior:	2,2,2	2
I am confident to explain the robot's behavior to someone else:	5,2,2	3

Table B2: Questionnaire 1, Control Group

	Scores	Average
I think I was able to fully reproduce the robot's behavior:	6,4,1	3.66
The robot's explanations from the first session helped me in this session:	2,1,1	1.33
I feel generally confident about collaborating with a robot that can explain its decision making to me:	7,4,1	4.00

Table B3: Questionnaire 2, Test Group

	Scores	Average
I think I was able to fully reproduce the robot's behavior:	3,2,2	2.33
I think it would be beneficial if the robot would be able to explain its decision making to me:	7,6,6	6.33

Table B4: Questionnaire 2, Control Group

	Scores	Average
Session Duration in seconds:	471, 378, 255	368
Session Duration in steps:	429,103,31	188
Number of explanations requested:	30,23,19	24
Jaccard Distance:	0.77,0.45,0.23	0.49
Correct Samples out of 49:	30,23,16	23

Table B5: Session 2 Performance Metrics Test Group

	Scores	Average
Session Duration in seconds:	898, 294, 284	492
Session Duration in steps:	433,368,296	365.66
Jaccard Distance:	0.83,0.49,0.40	0.57
Correct Samples out of 49:	34,25,22	27

Table B6: Session 2 Performance Metrics Control Group